

UNIVERSITY OF TARTU  
Institute of Computer Science  
Software Engineering Curriculum

Tõnis Kasekamp

A Web Application to Support  
Researchers in Predictive Process  
Monitoring Tasks

Master's Thesis (30 ECTS)

Supervisor: Fabrizio Maria Maggi

Tartu 2018

## **A Web Application to Support Researchers in Predictive Process Monitoring Tasks**

**Abstract:** Predictive Process Monitoring aims at predicting the outcome, time or cost of an ongoing execution of a business process using past process executions recorded in event logs. In this Master's Thesis, we describe the functionality of a web application (Nirdizati Research) that can be used to find a predictive model extracted from a given event log that can be used at runtime for making predictions on ongoing cases.

Nirdizati Research allows the user to create a model to predict the remaining time, the outcome and the next activity of an ongoing process execution. The application offers various configuration options in a way that different predictive models can be rated using various techniques and methods. There are also options to compare the created prediction models using different metrics. The tool has been evaluated on an authentic event log concerning the treatment process of sepsis patients in a hospital. Furthermore, the performance of the tool has been measured using a real-life log pertaining to the application process in a financial institute.

**Keywords:** Predictive Process Monitoring, Process Mining, Process Analytics Tool

**CERCS: P170** - Computer Science, Numerical Analysis, Systems, Control

## **Veebipõhine tööriist ennetava protsessijälgimise uurimise toetamiseks**

**Lühikokkuvõte:** Äriprotsesside ennustav seire üritab ennustada hetkel toimuva äriprotsessi lõpptulemust, kestvust või maksumust, kasutades selleks sündmuste logides leiduvaid eelnenud sündmuseid. Käesolevas magistritöös kirjeldame veebirakenduse funkionaalsust (Nirdizati Research), mis võimaldab kasutajatel leida konkreetse sündmuste logi jaoks parima ennustava mudeli. Seda mudelit saab seejärel kasutada käimasolevate protsessidele ennustuste tegemiseks.

Nirdizati Research lubab kasutajal teha mudeli äriprotsessi järele jäänud aja, tulemuse või järgmise sündmuse ennustamiseks. Rakendus pakub mudeli tegemiseks erinevaid konfigureerimise võimalusi ning võimaldab visuaalselt võrrelda genereeritud mudeleid. Rakendust on hinnatud haigla raviprotsessi sündmuste logi põhjal. Jõudluse hindamiseks on kasutatud finantsinstitutsiooni taotlusprotsessi logifaili.

**Võtmesõnad:** Protsessi ennustav seire, Protsessikaeve, Protsessi analüüsi tööriistad

**CERCS: P170** - Arvutiteadus, arvanalüüs, süsteemid, juhtimine

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Related work</b>	<b>6</b>
<b>3</b>	<b>Background</b>	<b>7</b>
3.1	Event log . . . . .	7
3.2	Predictive Process Monitoring . . . . .	7
3.2.1	Building the model . . . . .	7
3.2.2	Runtime predictions . . . . .	9
3.3	Hyperparameter optimization . . . . .	9
3.4	Nirdizati Training . . . . .	9
3.4.1	Implementation . . . . .	10
<b>4</b>	<b>Contribution</b>	<b>12</b>
4.1	Event log splitting . . . . .	12
4.2	Encoding . . . . .	13
4.2.1	Encoding methods . . . . .	14
4.2.2	Encoding configuration . . . . .	16
4.2.3	Task generation type . . . . .	17
4.3	Labelling . . . . .	18
4.3.1	Classification label types . . . . .	18
4.3.2	Regression label types . . . . .	20
4.4	Temporal and inter-case features . . . . .	21
4.5	Clustering methods . . . . .	21
4.6	Classification methods . . . . .	22
4.7	Regression methods . . . . .	22
4.8	Evaluation metrics . . . . .	22
4.8.1	Classification metrics . . . . .	22
4.8.2	Regression metrics . . . . .	23
<b>5</b>	<b>Functionality overview</b>	<b>25</b>
5.1	Log upload page . . . . .	26
5.2	Log details page . . . . .	27
5.3	Splitting page . . . . .	28
5.4	Labelling page . . . . .	29
5.5	Task status page . . . . .	31
5.6	Training page . . . . .	32
5.6.1	Classification methods . . . . .	33
5.6.2	Regression methods . . . . .	34

5.6.3	Clustering methods . . . . .	35
5.6.4	Encoding methods . . . . .	35
5.6.5	Labelling . . . . .	35
5.6.6	Temporal and inter-case features . . . . .	36
5.6.7	Hyperparameter optimization . . . . .	36
5.7	Validation page . . . . .	37
5.7.1	Classification results . . . . .	38
5.7.2	Regression results . . . . .	40
<b>6</b>	<b>Tool implementation</b>	<b>42</b>
6.1	Architecture . . . . .	42
6.1.1	Back-end architecture . . . . .	43
6.1.2	Front-end architecture . . . . .	43
6.2	Technologies . . . . .	44
6.2.1	Back-end technologies . . . . .	44
6.2.2	Front-end technologies . . . . .	45
6.2.3	Development process . . . . .	45
<b>7</b>	<b>Evaluation and Comparison</b>	<b>46</b>
7.1	Evaluation . . . . .	46
7.1.1	Remaining time prediction . . . . .	46
7.1.2	Next activity prediction . . . . .	48
7.2	Performance . . . . .	49
7.2.1	Encoding methods . . . . .	50
7.2.2	Log size impact on encoding performance . . . . .	50
7.2.3	Machine learning methods . . . . .	51
7.2.4	Hyperparameter optimization . . . . .	52
7.2.5	Comparison with Nirdizati Training . . . . .	53
7.3	Comparison with Nirdizati Training . . . . .	54
7.3.1	New features . . . . .	54
7.3.2	Improvements . . . . .	54
<b>8</b>	<b>Conclusion</b>	<b>56</b>

# 1 Introduction

Business process executions can be supported by information systems. These systems store events occurring during process executions in an *event log*, which can then be used as a source of information to improve future executions. The discipline of extracting knowledge from event logs is called process mining. The information in event logs can be used to discover the actual process model from the flow of events (process discovery), check the log against an existing process model (conformance checking) and to improve an already existing process model (model enhancement) [1].

A subset of process mining is Predictive Process Monitoring, which aims at answering questions about the future of processes which are currently executing. These predictions can be used to fulfill business goals. For example, Predictive Process Monitoring techniques could be used by an insurance company to first predict the time it takes to internally process an insurance application and then to allocate more resources for executions that are predicted to be slow so that they can be completed faster.

One of the challenges in Predictive Process Monitoring is identifying a predictive model that is best suited for a particular event log. This predictive model is built starting from the log, but the creation of the model can be configured with a multitude of options that can strongly affect its accuracy. However, tuning these options can be a time-consuming task if done manually.

In this thesis we aim at answering questions such as: "What is the best regression method for predicting the remaining time of a process?", "Does the event when the prediction is given affect the quality of the prediction?", "Can hyperparameter optimization provide a better accuracy than choosing the parameters manually?" and "Do inter-case features improve the quality of the predictive model?".

To answer these questions, we present a Web Application to Support Researchers in Predictive Process Monitoring Tasks called Nirdizati Research. The application helps researchers test various configuration options and compare the generated models through various validation metrics. The tool has been evaluated using a real-life event log concerning a treatment process in a hospital. The evaluation includes an use case of how the application could be used by researchers in a real-world context. The functionality and performance of the tool was compared against a similar application Nirdizati Training.

## 2 Related work

Existing predictive process monitoring prediction types can be classified as concerning numerical, categorical or a future sequence of events [2]. Numerical predictions are usually related to the duration or the cost of the ongoing case. Categorical predictions can be about the risk or the outcome of an ongoing case. Activity sequence predictions try to estimate the next activities of an ongoing case.

The presented tool considers a subset of numerical and categorical predictions, which are further described in Section 4.3.

One of the numerical predictions is **Remaining Time**, which aims at answering questions such as "How long will it take for a case to be completed?" and "What will be the duration of the currently executing activity?". In [3] the authors offer to answer these questions using past cases of event logs. They propose to use non-parametric regression to calculate the remaining case execution time. This method is shown to be superior to deducting the already elapsed cycle time from the average cycle time of the case. In [4] the authors present a way to use annotated transition systems to determine the duration of a case. Another method demonstrated in [5] is to use Petri nets to predict the remaining time of a process execution.

Categorical or **Outcome Based** predictions provide a prediction from a limited set of values. These methods try to answer questions such as "What is the next activity of an ongoing case?" and "What is the outcome of a given case?", "What is the value of a trace attribute of an ongoing case?". One method presented in [6] proposes a way to predict the outcome with rules defined with linear temporal logic. In [7] the authors predict the outcome of a trace by clustering the trace prefixes based on control flow information and separately classifying prefixes of different clusters. During a trace execution, the ongoing trace is compared against the classifier in the corresponding cluster to provide a prediction.

Predicting a sequence of future activities is a more recent addition to prediction types and this type of prediction is not included in this thesis. A method presented in [8] uses an annotated data-aware transition system to make a prediction about future activities. In [9], the authors propose to use past process executions in combination with a-priori knowledge to make a prediction.

## 3 Background

### 3.1 Event log

Processes in the real world can be described as a series of events with attributes such as the event name and the time of execution. These events can be grouped together to form a trace, which is the full execution of a given process case. Multiple traces can be grouped into an event log.

Each event and trace can have various attributes. Event attributes are most commonly the name of the event and the timestamp, but there can be additional string or numerical attributes. Trace attributes apply for all events in a given trace.

The **prefix** of a trace refers to a subsequence of events up to a given index or **prefix length**. For example, a prefix length of 3 indicates all events up to the 3rd event in a trace.

Log files are usually stored in the eXtensible Event Stream (XES) or MXML formats, which are both based on the XML schema. A subset of a real-life XES log [10] is shown in Listing 1. The example shows three events about a treatment process with various attributes, including the most common attributes `concept:name` and `time:timestamp`.

### 3.2 Predictive Process Monitoring

Predictive Process Monitoring is a family of methods for giving predictions and recommendations about currently running cases. Predictive Process Monitoring works first by building a predictive model from historical process executions, which are described in an event log. Predictions are then continuously provided to the user using this model over ongoing cases. The predictions can be about the remaining time, the outcome or the next activity of a trace, or about the value of any other attribute.

#### 3.2.1 Building the model

Building a predictive model for a process means finding a set of parameters for a machine learning method so that if presented with a partial or on-going trace, it can make an accurate prediction. The partial traces can have various lengths and choosing a prefix length helps us make predictions at different points in the process execution. For example, a prediction for the process remaining time is very different at the start of the trace than near the end of the trace. Therefore a different model should be created for every prefix length.

---

```

<trace>
  <string key="concept:name" value="H"/>
  <event>
    <boolean key="InfectionSuspected" value="false"/>
    <string key="org:group" value="A"/>
    <boolean key="DiagnosticBlood" value="false"/>
    <boolean key="DisfuncOrg" value="false"/>
    <boolean key="SIRSCritTachypnea" value="true"/>
    <boolean key="Hypotensie" value="false"/>
    <boolean key="SIRSCritHeartRate" value="true"/>
    <boolean key="Infusion" value="false"/>
    <boolean key="DiagnosticArtAstrup" value="false"/>
    <string key="concept:name" value="ER Registration"/>
    <int key="Age" value="80"/>
    <boolean key="DiagnosticIC" value="false"/>
    <boolean key="DiagnosticSputum" value="false"/>
    <boolean key="DiagnosticLiquor" value="false"/>
    <boolean key="DiagnosticOther" value="false"/>
    <boolean key="SIRSCriteria2OrMore" value="false"/>
    <boolean key="DiagnosticXthorax" value="false"/>
    <boolean key="SIRSCritTemperature" value="false"/>
    <date key="time:timestamp" value="2014-03-11T09:50:02.000+01:00"/>
    <boolean key="DiagnosticUrinaryCulture" value="false"/>
    <boolean key="SIRSCritLeucos" value="false"/>
    <boolean key="Oligurie" value="false"/>
    <boolean key="DiagnosticLacticAcid" value="false"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="Diagnose" value="G"/>
    <boolean key="Hypoxie" value="false"/>
    <boolean key="DiagnosticUrinarySediment" value="false"/>
    <boolean key="DiagnosticECG" value="false"/>
  </event>
  <event>
    <string key="org:group" value="C"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="concept:name" value="ER Triage"/>
    <date key="time:timestamp" value="2014-03-11T09:51:06.000+01:00"/>
  </event>
  <event>
    <string key="org:group" value="A"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="concept:name" value="ER Sepsis Triage"/>
    <date key="time:timestamp" value="2014-03-11T09:51:26.000+01:00"/>
  </event>

```

---

Listing 1: Trace of an event log in XES format [10]

Predictive Process Monitoring always starts with an event log. The first step is to split the encoded dataset into a training and test set. The training set will be used to build the predictive model while the test set will be used to validate quality of the predictive model.

To be able to apply prediction algorithms, the event log must be transformed or encoded into a suitable format. The five encoding methods used in this thesis were presented in a paper by Leontejeva et. al. [11]. The log file is encoded up to a specified prefix length.

The next step after the encoding is labelling the dataset. The value of the label is the value we are trying to predict. The label can be a numeric value, in which case regression methods are used, or a categorical value, in which case classification methods are used. Some label values are dependent on the prefix length, such as the remaining time.

The encoded traces are then given as input to an algorithm which will create the predictive model. There are several ways to configure how the predictive model is generated, including the choice of classification/regression algorithm, the number of events or prefixes to consider in a trace and the choice of the encoding method.



Additional features can be included in the encoded log file to improve the accuracy of the predictions.

The encoded traces can be clustered so that the predictive algorithm is applied to each cluster. For a given prefix length we create different clusters of similar prefixes and for each cluster we create a different regressor/classifier. In this way when a prediction has to be made for a currently ongoing trace the current prefix is first associated to a cluster and then the corresponding regressor/classifier is queried. This is done to increase the accuracy of the classification/regression algorithm.

After the model has been created, it can be compared against other models using various evaluation metrics to find the most suitable for the prediction task.

### **3.2.2 Runtime predictions**

Making a prediction for an on-going process means using the model on a partial trace. The partial trace will be encoded as per the configuration used to build the model. If the model configuration includes clustering then the trace is associated to a corresponding cluster. The end result is a prediction for the on-going trace, for example how long it will take to complete, if the prediction goal is the remaining time.

## **3.3 Hyperparameter optimization**

Hyperparameter optimization refers to choosing the optimal set of input parameters for a machine learning method. For example, the number of neighbours for a k-neighbours classifier is a hyperparameter as it is set before the training process. For any given dataset, it can be difficult to know in advance the most suitable hyperparameters. The optimization process will try out various values for the parameters and return the best configuration for a selected performance metric. This process improves the accuracy of predictions and saves time for the researcher by finding the best hyperparameters without trying them out manually [12].

## **3.4 Nirdizati Training**

Nirdizati Training is a web-based tool that supports research into predictive process analysis. The core predictive methods and general functionality was presented in [13]. Later it was improved in [14] by adding a prediction task queue and various prediction result visualization options.

The front-end application allowed the user to upload a log file, configure the settings of the prediction tasks and to plot the results on various graphs.

The back-end application was divided into 6 modules besides the storage module. The Log Manager module stored and provided information about the logs in storage. The Encoder module transformed a log provided by the Log Manager into an encoded CSV file and stored it onto the disk. The Prediction module in turn retrieved the encoded file and created a predictive model. The last module in the pipeline was the Evaluation module, which stored, aggregated and provided the results of the prediction tasks. To allow for the execution of several prediction tasks in parallel, the calculations were orchestrated by the Queuing module which comprised of several worker applications. Figure 1 shows the architecture of Nirdizati Training.

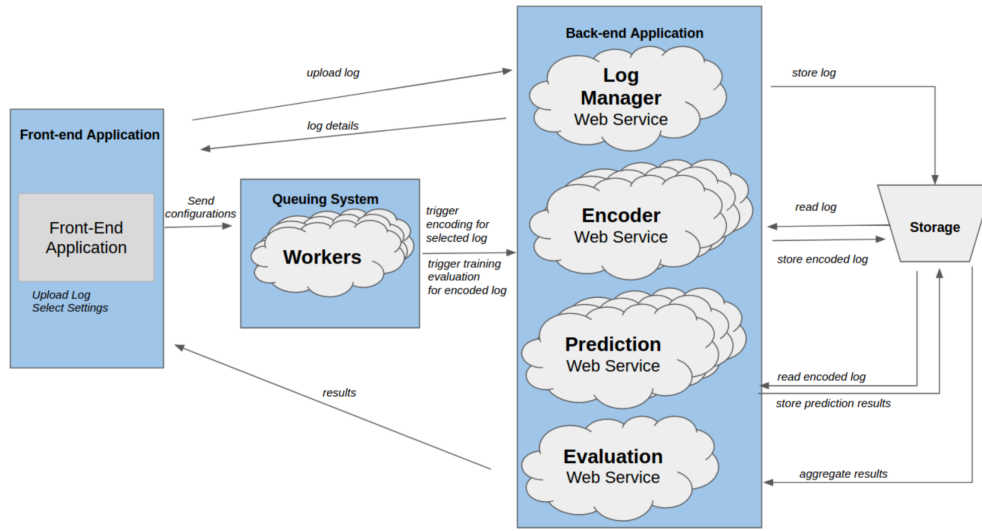


Figure 1: Nirdizati Training architecture

### 3.4.1 Implementation

The back-end application consisted of a web server and a queuing system. The Django<sup>1</sup> web server was written in Python 2 and it forwarded the prediction tasks from the front-end to the queuing system. The form to submit an outcome prediction task is shown in Figure 2.

The front-end application was developed using AngularJS with a Material theme<sup>2</sup>. The application allowed the user to upload a single log file in XES format for analysis, specify the configuration for outcome and remaining time prediction

<sup>1</sup><https://www.djangoproject.com/>

<sup>2</sup><https://material.angularjs.org/latest/>

Figure 2: Nirdizati Training outcome prediction options

tasks and see the results of the prediction tasks. Figure 3 shows the results page of the outcome prediction task.

Run	Fmeasure	AUC	ACC
1 KNN_simpleIndex_None_clustering (1)	1	1	1

Figure 3: Nirdizati Training outcome prediction results

The queuing system was implemented using Django-RQ<sup>3</sup>, which is a Redis<sup>4</sup> based queuing library specifically developed for use in Django projects. This system allowed the web application to serve content while completing multiple prediction tasks in the background.

<sup>3</sup><https://github.com/ui/django-rq>

<sup>4</sup><https://redis.io>

## 4 Contribution

The tool presented in this thesis is Nirdizati Research. The application is an improved version of Nirdizati Training which was developed in [13] and [14]. This section describes the core concepts and the server side functionalities of Nirdizati Research.

The application is deployed at <http://research.nirdizati.org/#/>. The source code of the front-end<sup>5</sup> and backend<sup>6</sup> applications are available in public Github repositories.

### 4.1 Event log splitting

In machine learning, a dataset is generally split into a training set for training the model and a test set for validating the trained model. The application offers four methods for splitting the event log: sequential ordering, random ordering, temporal ordering and strict temporal ordering. By default, the application designates 80% of the event log as the training set and 20% as the test set.

**Sequential ordering** is the default split type and it applies no sorting or ordering on the log file. The log file is split according to the percentage considering the traces in the order in which they are stored in the log.

**Random ordering** sorts the log file randomly before splitting it into training and test set.

**Temporal ordering** sorts the traces by the timestamp date of the first event in the trace. The sorted log file is then split according to the split percentage.

**Strict temporal ordering** is an advanced version of temporal ordering. It first uses the temporal ordering to create a training and a test set. Next, it filters the training set so that it includes only the traces where the last event ends before the start of the first trace in the test set. This ordering prevents the traces in the training and test sets from overlapping. However, because the training set is filtered, it might result in fewer traces with respect to other split types.

---

<sup>5</sup><https://github.com/TKasekamp/predict-react>

<sup>6</sup><https://github.com/TKasekamp/predict-python>

## 4.2 Encoding

Encoding means transforming the data stored in event logs into a shape that can be used by machine learning methods. The application supports the encoding methods introduced in [11], which are illustrated in Figure 4. These methods vary by the amount of data they retain from the event log, however all of them only take into account the control-flow of a trace.

consultation ultrasound ... payment label					
$\sigma_1$	1	1	...	0	false
...					
$\sigma_k$	0	0	...	1	true

(a) *boolean* encoding.

consultation ultrasound ... payment label					
$\sigma_1$	2	1	...	0	false
...					
$\sigma_k$	0	0	...	4	true

(b) *frequency-based* encoding.

event_1 ... event_m label			
$\sigma_1$	consultation	ultrasound	false
...			
$\sigma_k$	order rate	payment	true

(c) *simple index* encoding.

age event_1 ... event_m ... department_last label					
$\sigma_1$	33	consultation	ultrasound	...	nursing ward false
...					
$\sigma_k$	56	order rate	payment	...	clinic true

(d) *index latest payload* encoding.

Figure 4: Encoding methods [11]

In the following examples and in the application, the **Prefix length** denotes how many events to consider from a trace.

To illustrate the various encoding methods and configurations implemented in the application, we are going to introduce a sample event log in Tables 1 and 2. The event log pertains to an insurance claim procedure where each process has an assignee and each step has a cost for the company. The event log contains 4 traces and the event attributes are the Timestamp and a numeric attribute Cost. Table 1 defines trace attributes Priority and Assignee.

Case id	Priority	Assignee
1	15	Mike
2	40	Sam
3	20	Elsa
4	35	Mike

Table 1: Sample Event log trace attributes

Case id	Event name	Cost	Timestamp
1	A	50	19/04/2018 14:00:00
2	A	50	19/04/2018 15:00:00
1	B	100	19/04/2018 15:05:00
2	C	100	19/04/2018 15:07:00
3	B	50	20/04/2018 10:00:00
3	C	400	20/04/2018 14:00:00
4	D	50	21/04/2018 11:00:00
4	D	100	21/04/2018 11:10:00
1	D	200	24/04/2018 14:30:00
1	E	200	24/04/2018 14:32:00
3	F	100	03/05/2018 10:00:00
2	D	200	04/05/2018 9:05:00
4	F	200	08/05/2018 14:30:00
3	D	200	12/06/2018 14:01:00
3	E	200	21/06/2018 11:00:00

Table 2: Sample event log

#### 4.2.1 Encoding methods

The application supports a total of five encoding types: simple index, last payload, complex, boolean, and frequency. The examples are created using the log defined in Table 2. All examples are with prefix length 2, meaning that only the first two events in a trace are considered. The log files are described here without a label.

**Simple index** encoding encodes the log file by inserting the event name at each prefix position up to prefix length. An example of simple index encoding with prefix length 2 is shown in Table 3.

case_id	prefix_1	prefix_2
1	A	B
2	A	C
3	B	C
4	B	D

Table 3: Simple index encoding with prefix length 2

**Last payload** index encoding works similar to simple index encoding as the event names are included at every prefix. However, it also includes the event attributes of the event at the last prefix, in this case the Cost attribute. An example of last payload encoding with prefix length 2 is shown in Table 4.

case_id	prefix_1	prefix_2	Cost
1	A	B	100
2	A	C	100
3	B	C	400
4	D	D	100

Table 4: Last payload encoding with prefix length 2

**Complex** index encoding is another index-based encoding that includes all event attributes at every considered prefix length. An example of complex index encoding with prefix length 2 is shown in Table 5.

case_id	prefix_1	Cost_1	prefix_2	Cost_2
1	A	50	B	100
2	A	50	C	100
3	B	50	C	400
4	D	50	D	100

Table 5: Complex index encoding with prefix length 2

**Boolean** encoding creates a column for each activity name in the event log and the value is true if the event has occurred in the case up until the specified prefix length. Trace attributes are not considered. An example of boolean encoding with prefix length 2 is shown in Table 6.

case_id	A	B	C	D	E	F
1	true	true	false	false	false	false
2	true	false	true	false	false	false
3	false	true	true	false	false	false
4	false	false	false	true	false	false

Table 6: Boolean encoding with prefix length 2

**Frequency** encoding is similar to Boolean encoding, but it uses the number of occurrences of each event. An example of frequency encoding with prefix length 2 is shown in Table 7.

case_id	A	B	C	D	E	F
1	1	1	0	0	0	0
2	1	0	1	0	0	0
3	0	1	1	0	0	0
4	0	0	0	2	0	0

Table 7: Frequency encoding at prefix length 2

#### 4.2.2 Encoding configuration

In addition to the encoding method, the encoding of the log file can be configured with the padding type and the prefix length. These options can be used with all encoding methods. The following encoding options are described with simple index encoding and using the event log defined in Table 2.

Prefix length of 3 means that only the first 3 events in a case are considered. Using a prefix length of 3 will encode the log as follows.

```

A  B  D
A  C  D
B  C  F
D  D  F

```

The number of events in a case can vary greatly, but the data mining methods require that each case in the encoded log file has the same number of columns. There are two strategies on how to handle this: discarding cases that have fewer events than the selected prefix length or to pad the encoded log with 0 values.

Which strategy to use depends on each specific log file and prefix length. Using the "no padding" option might mean that the training and test set contain too few cases to make a meaningful prediction. Using the "zero padding" option might mean that there are so many zero values that they distort the generated model.

When using the option "no padding" and a prefix length of 5, the encoded log file will only consist of case 3 as the other cases have been discarded.

```

B  C  F  D  E

```

When using the option "zero padding" and a prefix length of 5, the encoded log file will consist of all 5 cases.

```

A  B  D  E  0
A  C  D  0  0
B  C  F  D  E
D  D  F  0  0

```



### 4.2.3 Task generation type

Nirdizati Research also provides a macro option for creating many prediction tasks with different prefix length. This option is called "Task generation type" and the options are to create only one task with only this prefix length, multiple tasks with every prefix length from 1 up to the specified value or with every prefix length up to the specified value in a single log file.

Using the log file defined in Table 2 with the configuration "no padding", task generation type "up to prefix length" and prefix length 5, there will be a total of 5 encoded log files with the following content.

Prefix length 1

A  
A  
B  
D

Prefix length 2

A B  
A C  
B C  
D D

Prefix length 3

A B D  
A C D  
B C F  
D D F

Prefix length 4

A B D E  
B C F D

Prefix length 5

B C F D E

The previous example created 5 separate log files with different prefix length. Using the task generation option "all in one", padding "zero padding" and prefix length 5, all these will be put into a single log file. The padding option here refers to the total length of the log. If it was set to "no padding", only the traces that are at least as long as the prefix length would be included.

Prefix length 5 with zero padding

```

A  0  0  0  0
A  B  0  0  0
A  B  D  0  0
A  B  D  E  0
A  0  0  0  0
A  C  0  0  0
A  C  D  0  0
B  0  0  0  0
B  C  0  0  0
B  C  F  0  0
B  C  F  D  0
B  C  F  D  E
D  0  0  0  0
D  D  0  0  0
D  D  F  0  0

```

Prefix length 5 with no padding

```

B  0  0  0  0
B  C  0  0  0
B  C  F  0  0
B  C  F  D  0
B  C  F  D  E

```

### 4.3 Labelling

A dataset label is the output value which the machine learning algorithm is trying to predict. This label is numeric for regression methods and a categorical value for classification methods. The presented application supports two labelling types for regression methods and four labelling types for classification methods. Examples are provided for all labelling types.

#### 4.3.1 Classification label types

The classification methods label types and their differences are outlined in Table 8.

Label type	Classification type	Choose threshold	Choose trace attribute	Depends on prefix length
Duration	Binary	Yes	No	No
Numerical attribute	Binary	Yes	Yes	No
Next activity	Multiclass	No	No	Yes
String attribute	Multiclass	No	Yes	No

Table 8: Classification label types

**Binary classification** means that each instance in the training set can belong to one of two classes. Nirdizati Research supports two labelling types for binary

classification: Duration and Trace numerical attribute. Both of these numerical attributes are classified as **True** or **False**, depending on if they are below or above a certain threshold. The threshold can be the mean of the values of the attribute column in the entire training set or a custom threshold specified by the user. Both of the labelling types are the same across all prefix lengths in a trace.

Duration is the time in seconds from the first to the last event in a trace. The threshold is the number below which traces are classified as "Fast" or **True**. Traces with duration greater than the threshold are classified as "Slow" or **False**.

Traces can contain additional meta data. If these values are numerical, then these values can be classified as above or below a certain threshold.

case_id	prefix_1	label
1	A	<b>True</b>
2	A	<b>True</b>
3	B	<b>False</b>
4	D	<b>True</b>

Table 9: Duration label

case_id	prefix_1	label
1	A	<b>True</b>
2	A	<b>False</b>
3	B	<b>True</b>
4	D	<b>False</b>

Table 10: Numerical attribute label

Examples of duration and numerical attribute labelling are outlined in Tables 9 and 10. Both examples use simple index encoding with prefix length 1 with the event log defined in Table 2.

The calculated duration values at prefix length 1 are equivalent to the remaining time values in Table 14. The threshold type is the mean value and the calculated threshold is 2,137,605.

The numeric attribute is Priority with threshold type mean value 27.5.

**Multiclass classification** means that the label of each instance can belong to one of three or more classes. Nirdizati Research supports two labelling types for multiclass classification: Next activity and Trace string attribute.

Next activity aims at predicting the event that will happen next at this prefix length. The label value is dependent on the prefix length as a different event will occur next for any given prefix in the trace. The user does not need to choose any other options to use next activity labelling.

Any trace attribute which is not a number can be used as a source for a multiclass classification. As a trace attribute, the label will be the same across all prefix lengths.

Examples of next activity and string attribute labelling are shown in Tables 11 and 12. Both examples use simple index encoding with prefix length 1 using the event log defined in table 2. String labelling uses the attribute Assignee.

case_id	prefix_1	label
1	A	B
2	A	C
3	B	C
4	D	D

Table 11: Next activity label

case_id	prefix_1	label
1	A	Mike
2	A	Sam
3	B	Elsa
4	D	Mike

Table 12: String attribute label

#### 4.3.2 Regression label types

The application supports the trace remaining time and the trace numerical attribute as a regression label. The differences of the regression labels are outlined in Table 13.

Label	Choose trace attribute	Depends on prefix length
Remaining time	No	Yes
Numerical attribute	Yes	No

Table 13: Regression label types

For an event at a given prefix length in a trace, the **remaining time** is the time in seconds until the last event in a trace. The value changes depending on the prefix length.

case_id	prefix_1	label
1	A	433,920
2	A	1,274,700
3	B	5,360,400
4	D	1,481,400

Table 14: Remaining time label

case_id	prefix_1	label
1	A	15
2	A	40
3	B	20
4	D	35

Table 15: Numerical attribute label

Any trace **numerical attribute** can also be used as a label. This label will be the same for any selected prefix length.

Examples of remaining time and numerical attribute labelling are shown in tables 14 and 15. Both examples use simple index encoding with prefix length 1 using the event log defined in table 2. Numerical labelling uses the attribute Priority.

## 4.4 Temporal and inter-case features

The encoded log file can be supplemented with additional feature columns, which can improve the accuracy of the generated model [15][16]. All the presented features are numerical values and are dependent on the selected prefix length. They can be classified as temporal and inter-case features.

The supported temporal features are the **remaining time** and **elapsed time**. These features are intra-case, meaning that their value depends on the execution of the specific trace [15]. The remaining time is the time in seconds from the event at the current prefix length to the last event in the trace. The elapsed time is the time in seconds from the first event in the trace to the event at the current prefix length.

Inter-case features are created using aggregate metrics over all the executed events in the event log [15]. The three supported inter-case features are **executed events**, **resources used** and **new traces** concerning the specified feature in the time window corresponding to the day in which the last event of the current prefix occurred. Using the sample event log defined in Table 2, an example of inter-case feature columns using simple index encoding with prefix length 1 is shown in Table 16.

case_id	prefix_1	executed_events	resources_used	new_traces
1	A	4	4	2
2	A	4	4	2
3	B	2	2	1
4	D	2	2	1

Table 16: Inter-case features

## 4.5 Clustering methods

Clustering is a machine learning technique which aims at improving the prediction accuracy by grouping together similar elements and learning a predictive model from each group separately. The presented application allows the user to choose between no clustering and the **k-means** clustering method.

**k-means** is a popular clustering method for data mining. k-means divides the  $n$  observations in an unlabelled dataset into  $k$  clusters so that each observation belongs to a cluster with the nearest mean. The algorithm then tries to group the observations in such a way that the the distance between each object in a is minimal.

## 4.6 Classification methods

Classification methods are used in machine learning to predict a categorical value. The application supports 3 classification methods: **Decision trees**, **Random Forest** and K-Nearest Neighbor (**KNN**).

A **Decision tree** is a graph where each branch split represents a decision and each end node represents the outcome of a test. The classification rules are generated from the path from the root to the end node.

**Random Forest** is a machine learning method that uses multiple decisions trees to make a prediction. The results of the decision trees are averaged to control over-fitting and improve the prediction accuracy.

The K-Nearest Neighbor (**KNN**) method predicts the value based on the k closest neighbors in the data set.

## 4.7 Regression methods

Regression methods are used in machine learning to predict a numeric value. The application supports 3 regression methods: **Linear**, **Lasso** and **Random forest**.

**Linear** regression fits a linear function between one or many independent variables to make a prediction.

**Lasso** or the Least Absolute Shrinkage and Selection Operator decreases the value of some features to improve prediction accuracy [17].

**Random Forest** differs from the classification variant by returning a numeric value.

## 4.8 Evaluation metrics

In machine learning, a training set is used to fit a model and this model is then compared against the test set. The quality of the model can be represented with a variety of evaluation metrics.

### 4.8.1 Classification metrics

For classification methods, the application calculates the **Accuracy**, **Area Under the Curve** (AUC), **Precision**, **Recall** and **F1 score** evaluation metrics. When using the clustering option, the average value across all clusters is calculated. Binary classification tasks will also calculate the metrics presented in table 17.

**Accuracy** measures the share of correct predictions out of all predictions. This measure is calculated for both binary and multi-class classification tasks. The formula of accuracy is:

	Actual positive	Actual negative
Predicted positive	True positive (TP)	False positive (FP)
Predicted negative	False negative (FN)	True negative (TN)

Table 17: Confusion matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

**Precision** is the share of true positive values out of all predicted positive values. For multiclass classification, this metric is calculated by a function defined in `sklearn.metrics`<sup>7</sup> with the "macro" labelling average parameter. The formula is:

$$Precision = \frac{TP}{TP + FP}$$

**Recall** measures the share of correct predictions out of all positive predictions. For multiclass classification, this metric is calculated by a function defined in `sklearn.metrics`<sup>8</sup> with the "macro" labelling average parameter. The formula is:

$$Recall = \frac{TP}{TP + FN}$$

**F1-score** or F-measure is the weighted average of recall and precision. For multiclass classification tasks, this metric is calculated by a function defined in `sklearn.metrics`<sup>9</sup> with the "macro" labelling average parameter. The formula is:

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

**Area Under the Curve** or AUC calculates the area under the Receiving Operating Characteristic curve. This is calculated using the method provided by `sklearn.metrics`<sup>10</sup>.

#### 4.8.2 Regression metrics

For regression methods, the application calculates the **Root Mean Square Error** (RMSE), **Mean Absolute Error** (MAE) and **Coefficient Of Determination** (r-score) evaluation metrics. When using the clustering option, the average value across all clusters is calculated.

<sup>7</sup>[http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision\\_score.htm](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.htm)

<sup>8</sup>[http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall\\_score.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html)

<sup>9</sup>[http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\\_score.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)

<sup>10</sup><http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html>

**Root Mean Square Error** measures the difference between predicted and actual values. The formula is:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=0}^n (x_i - \hat{x}_i)^2}$$

where  $n$  is the number of rows in the dataset,  $x$  is the predicted value at  $x_i$  and  $\hat{x}_i$  is the actual value.

**Mean Absolute Error** measures the difference between predicted and actual values. The formula is:

$$MAE = \frac{1}{n} \sum_{i=0}^n |x_i - \hat{x}_i|$$

where  $n$  is the number of rows in the dataset,  $x$  is the predicted value at  $x_i$  and  $\hat{x}_i$  is the actual value.

**Coefficient Of Determination** is the square of the measure of the linear correlation between predicted and actual values.



## 5 Functionality overview

Nirdizati Research is a tool to find the most suitable predictive model for an event log. The general flow to use the application is as follows.

The first step is to upload a log file. The next step is to create a training/test split configuration using the event log. The Split configuration can be re-used for multiple prediction and labelling tasks.

Before creating a classification prediction task, there is an option to test out the labelling distribution. The configuration of the labelling tasks can then be used as an input for the classification tasks.

Classification and regression tasks can be created with a multitude of configuration options. When the tasks have been submitted, they are enqueued and will be completed when computing resources become available. The completed models can be compared using several evaluation metrics with various visualization options.

The front page of the application provides a walk through of how to use Nirdizati Research, as can be seen from Figure 5.

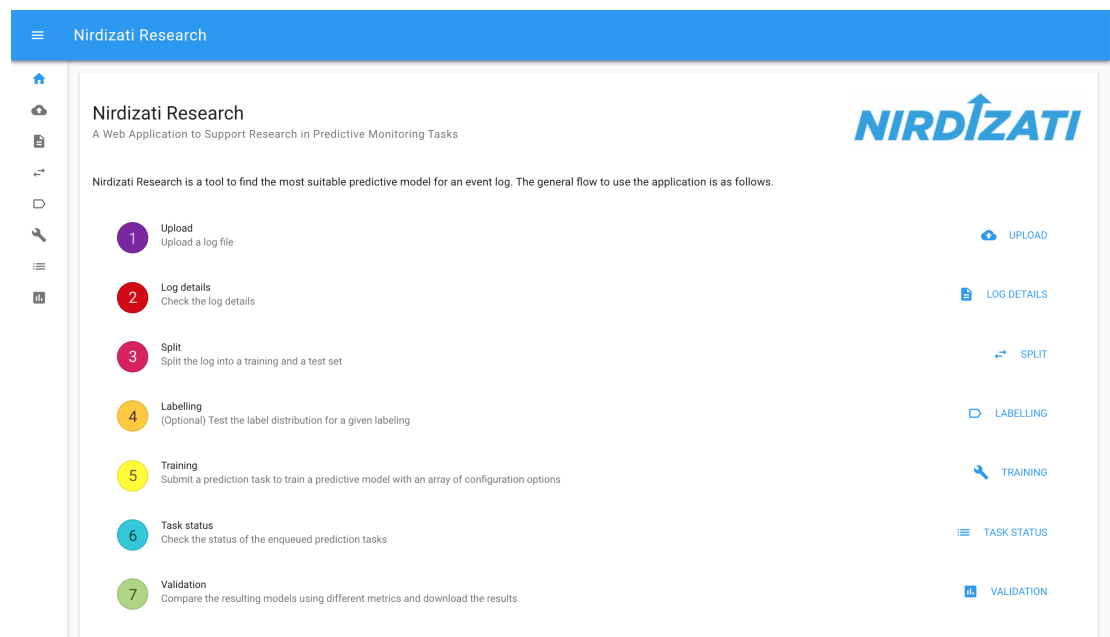


Figure 5: Nirdizati Research walkthrough

## 5.1 Log upload page

The upload page is used for uploading a log file. The user has two options when uploading the log file: it can be either a single log file or it can be two log files, where one is a training and the other is a test set. When uploading a single log file the user must specify how the training and test set are generated. This is configured in the splitting phase. Uploading two log files allows the user to fine-tune the contents of the training and test set. When uploading two log files the user can proceed to the Labelling or Training pages without creating the split.

The log files can be in plain and gzip compressed XES and MXML formats. During the upload process, a record of the log file is made in the database and the log file is processed to extract the data to create the charts on the Log details page. Figure 6 depicts the Log upload page.

The screenshot shows the 'Log upload page' of the 'Nirdizati Research' application. The page is divided into two main sections for file upload.

**Single log file upload**

After upload create a Split with this log!

Supported log file formats are **xes, mxml, xes.gz and mxml.gz**

Upload a single log file. Metrics for the charts on Log details page will be calculated during the upload and this process may take time. Remain patient!

No file chosen

**Training and test set upload**

Supported log file formats are **xes, mxml, xes.gz and mxml.gz**

Upload two log files. Metrics for the charts on Log details page will be calculated during the upload and this process may take time. Remain patient!

No file chosen

No file chosen

Figure 6: Log upload page

## 5.2 Log details page

The page features graphs that describe a log file, i.e, the number of events executed per day, the number of resources employed per day and the number of new cases started per day. Figure 7 shows the log details page with the events per day graph.

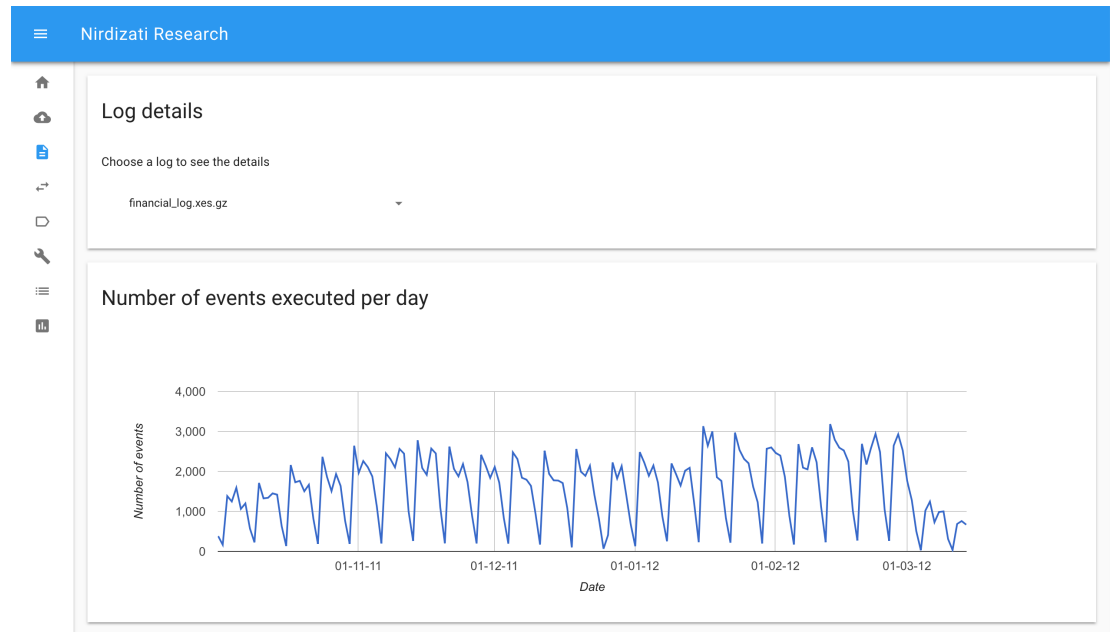


Figure 7: Logs page with events per day graph

### 5.3 Splitting page

The splitting page is used to split an uploaded log file into a training and a test set. A split is a single log file with a configuration for separating it into a training and test set. A split can also represent a training log and test log file, which are uploaded separately. This allows the user to reuse the same split configuration with multiple labelling and training tasks. The split create form can be seen in Figure 8.

To create a split, the user must select a log file, a split type and the training/test set percentage. The application supports four split types: sequential order, temporal order, random order and strict temporal order. The split types are discussed in more detail in section 4.1.

Figure 8: Split create form

After the splits have been created, the configuration can be verified in the split page tables, as can be seen in Figure 9.

id	Log	Split type	Test set %
1	general_example.xes	split_random	0.2
3	nonlocal.mxml.gz	split_sequential	0.2
4	financial_log.xes.gz	split_temporal	0.2
5	repairExample.xes	split_sequential	0.2

Logs uploaded as a separate training and test set.		
id	Training log	Test log
2	general_example_training.xes	general_example_test.xes

Figure 9: Split page tables

## 5.4 Labelling page

Nirdizati Research uses classification methods on an event log to predict the value of a given label. Due to the amount of configuration options, the application provides a page to try out different configurations for labelling types, prefix lengths and log padding types, which can be seen in Figure 10. This allows the user to see the distribution of labels in an encoded log file before applying data mining methods. For example, using the "no padding" option will mean that a high prefix length will decrease the amount of total traces in the training sets, leading to different label counts.

The screenshot shows the 'Labelling' configuration page in the Nirdizati Research application. The page has a blue header with the logo and a sidebar with navigation icons. The main content area is titled 'Labelling' and shows the file 'Split #1, general\_example.xes'. It contains two main sections: 'Encoded log padding' and 'Task generation type'. The 'Encoded log padding' section has two radio button options: 'No padding' (selected) and 'With 0 padding'. The 'Task generation type' section has three radio button options: 'Only this prefix length' (selected), 'Up to the prefix length', and 'All in one dataset'. A 'Prefix length (maximum 13)' input field is set to '1'. Below these sections, there is a 'Labelling' section with a description of the thresholding process. It includes a 'Label type' dropdown set to 'Trace duration' (Binary classification), a 'Threshold type' dropdown set to 'Label mean' (Threshold is label mean), and a 'Threshold (seconds)' input field set to '0'. At the bottom, there are 'SUBMIT' and 'RESET' buttons.

Figure 10: Label create form

The encoding options are discussed in sections 4.2.2 and 4.2.3. The classification labelling options are outlined in section 4.3.1. After selecting the labelling configuration and submitting the labelling task, the user is directed to the Tasks page. When the labelling task has been completed, the results can be viewed on the Labelling page.

The labelling task results can be filtered by the Split, the label type, the threshold type, the attribute name and prefix lengths. The filters can be seen in Figure 11. All available custom thresholds are available in a dropdown menu, so the user does not have to remember the exact threshold value.

Labelling result selection

Split #6, financial\_log.xes.gz

Label type

Trace number attribute

Binary classification

Threshold type

Custom

Use the threshold value below

Threshold

50000

Attribute name

AMOUNT\_REQ

Encoded log padding

☐ No padding
☒ With 0 padding

Figure 11: Label results filter

Below the filtering options there is a line chart that provides a high level overview of the count of labels. As can be seen from the next activity labelling results on an example log file in figure 12, the number and distribution of label classes change depending on the prefix length. The chart also shows that at prefix length 1 and 2 the next activity is always "Analyze Defect", making the prediction at this point trivial.

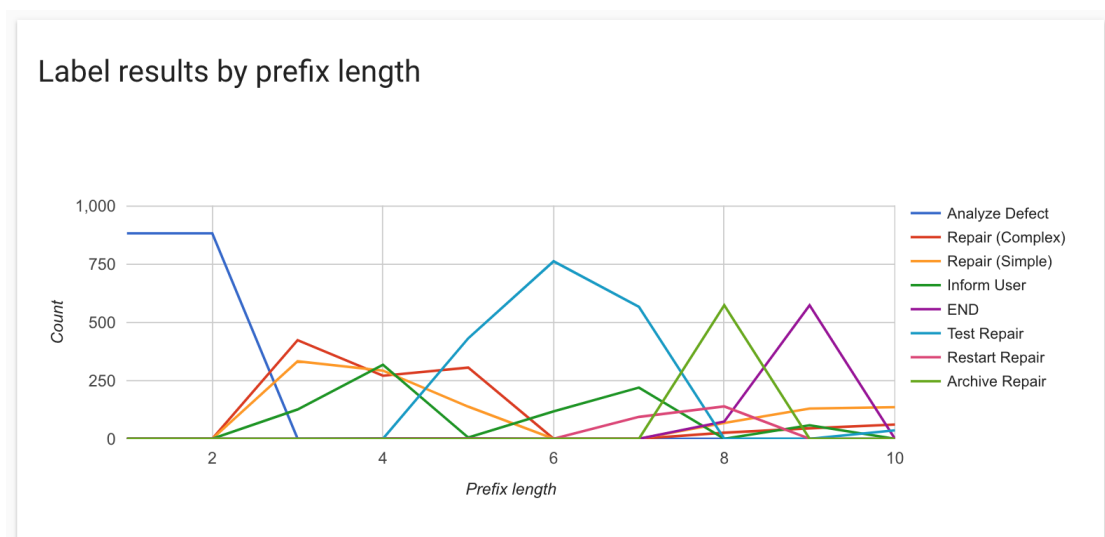


Figure 12: Next activity labels by prefix length

Next to the line chart is a bar chart that provides a more detailed overview of the label count distribution, as can be seen in figure 13. The "END" label in this case means the trace ends at this point, so there is no next activity.

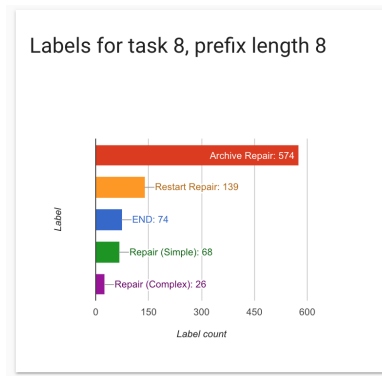


Figure 13: Next activity label details chart

## 5.5 Task status page

The Task status page provides an overview of all labelling, classification and regression tasks in the server. By default, this page automatically fetches tasks from the server every 10 seconds, but automatic fetching can be turned off. All tasks can be deleted. The full configuration of each task can be seen by clicking on the task table row. The tasks status page is shown in Figure 14.

Nirdizati Research								
Task status								
There are 16 tasks in the front-end application.								
REFRESH LIST	<input type="checkbox"/>	Automatically fetch tasks	<input checked="" type="checkbox"/>	Show completed tasks	<input checked="" type="checkbox"/>	Show delete button		
id		Status	Type	Created date	Modified date	Split	Error	Configuration
16	DELETE	completed	classification	5/12/2018, 4:50:19 PM	5/12/2018, 4:50:21 PM	Split #5, repairExample.xes		<a href="#">root</a>
15	DELETE	completed	regression	5/12/2018, 4:50:05 PM	5/12/2018, 4:50:17 PM	Split #5, repairExample.xes		<a href="#">root</a>
14	DELETE	completed	regression	5/12/2018, 4:50:05 PM	5/12/2018, 4:50:15 PM	Split #5, repairExample.xes		<a href="#">root</a>
13	DELETE	completed	regression	5/12/2018, 4:50:05 PM	5/12/2018, 4:50:12 PM	Split #5, repairExample.xes		<a href="#">root</a>
12	DELETE	completed	regression	5/12/2018, 4:50:05 PM	5/12/2018, 4:50:10 PM	Split #5, repairExample.xes		<a href="#">root</a>
11	DELETE	completed	regression	5/12/2018, 4:50:05 PM	5/12/2018, 4:50:08 PM	Split #5, repairExample.xes		<a href="#">root</a>
10	DELETE	completed	labelling	5/12/2018, 4:49:19 PM	5/12/2018, 4:49:42 PM	Split #5, repairExample.xes		<a href="#">root</a>
9	DELETE	completed	labelling	5/12/2018, 4:49:19 PM	5/12/2018, 4:49:40 PM	Split #5, repairExample.xes		<a href="#">root</a>
8	DELETE	completed	labelling	5/12/2018, 4:49:19 PM	5/12/2018, 4:49:37 PM	Split #5, repairExample.xes		<a href="#">root</a>
7	DELETE	completed	labelling	5/12/2018, 4:49:19 PM	5/12/2018, 4:49:35 PM	Split #5, repairExample.xes		<a href="#">root</a>
Rows per page 10 1-10 of 16 < >								

Figure 14: Task status table

## 5.6 Training page

The training page is where the user creates tasks to apply machine learning methods on an event log. First the user must choose the Split that contains the training and test set. The next choice, depending on the prediction task type, is whether to use regression to predict numeric values or classification to predict categorical values. The application also offers a choice for encoding options, clustering methods, learning methods and labelling. An overview of the configuration options is shown in Figure 15.

The application uses default parameters for each machine mining method. However, users can change the selection of these parameters or find the best configuration using hyperparameter optimization. Additional temporal and inter-case features can be added to the encoded log file.

After choosing the all the required inputs, the prediction task will be visible on the Task status page.

Nirdizati Research

Training Split #1, general\_example.xes

**Prediction method**

☒ Regression ☐ Classification  
Numeric values Categorical values

**Regression methods**

☒ Linear ☐ Random forest ☐ Lasso

**Clustering methods**

☒ None ☐ Kmeans clustering  
No clustering and train a single model  
Assign traces to k-means clusters and train a model for each cluster

**Encoding methods**

☒ Simple index ☐ Boolean ☐ Frequency ☐ Complex ☐ Last payload  
Each feature corresponds to a position in the trace and the possible values for each feature are the activity names. Event attributes are discarded.  
Features represent whether or not a particular activity has occurred in the trace. Event attributes are discarded.  
Features represent the absolute frequency of each possible activity. Event attributes are discarded.  
Each feature corresponds to a position in the trace and the possible values for each feature are the activity names and event attributes  
Features represent the event attributes of the last event that occurred in the trace.

**Encoded log padding**

☒ No padding ☐ With 0 padding  
Traces with length less than the specified prefix length will be discarded  
Traces with length less than the specified prefix length will be padded with 0

**Task generation type**

☒ Only this prefix length ☐ Up to the prefix length  
Create multiple tasks from the specified prefix length 1 up to this value

The maximum prefix length is 18. Values above maximum length might have inconsistent results.

Prefix length \*

1

Labelling

Temporal and intercase features

Hyperparameter Optimization

Linear regression

☐ Create and save models for runtime prediction

SUBMIT RESET

Figure 15: Training page

Multiple prediction tasks can be generated all together. A task will be created for each selected learning method, clustering method, encoding method and prefix length. By choosing all available options with hyperparameter optimization, the user can create all possible prediction tasks with the most suitable configuration by clicking the "Submit" button only once.



### 5.6.1 Classification methods

The application supports 3 classification methods: **Decision trees**, **Random Forest** and K-Nearest Neighbor (**KNN**). These methods are described in section 4.6 while section 4.3.1 provides an overview of the supported the classification labels.

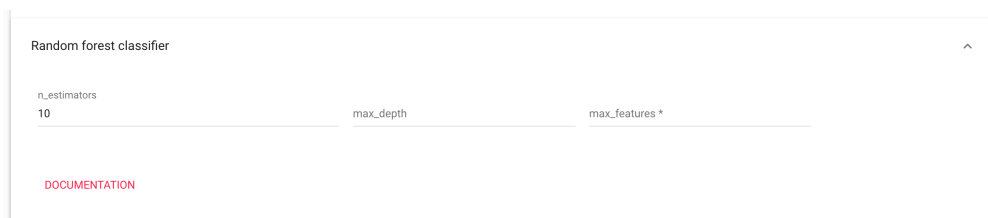
For **Decision tree**, the user can configure the maximum depth, the minimum sample split and the minimum sample split of the method. The configuration options are shown in Figure 16.



The screenshot shows a configuration panel for a "Decision tree classifier". It features three input fields: "max\_depth" with a value of 2, "min\_samples\_split \*" with a value of 2, and "min\_samples\_leaf \*" with a value of 1. Below these fields is a red "DOCUMENTATION" link. The panel has a title bar with an upward arrow and a collapse icon.

Figure 16: Decision tree configuration options

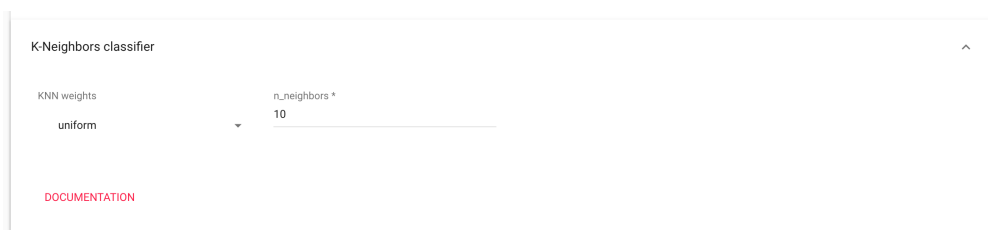
For **Random Forest**, the user can configure the number of estimators, the maximum depth and the maximum number of features. The configuration options are shown in Figure 17.



The screenshot shows a configuration panel for a "Random forest classifier". It features three input fields: "n\_estimators" with a value of 10, "max\_depth" with a value of 10, and "max\_features \*" with a value of 10. Below these fields is a red "DOCUMENTATION" link. The panel has a title bar with an upward arrow and a collapse icon.

Figure 17: Random forest configuration options

For K-Nearest Neighbor (**KNN**), the user can configure the KNN weights and the number of neighbors. The configuration options are shown in figure 18.



The screenshot shows a configuration panel for a "K-Neighbors classifier". It features two input fields: "KNN weights" with a dropdown menu set to "uniform", and "n\_neighbors \*" with a value of 10. Below these fields is a red "DOCUMENTATION" link. The panel has a title bar with an upward arrow and a collapse icon.

Figure 18: KNN configuration options

### 5.6.2 Regression methods

The application supports 3 regression methods: **Linear**, **Lasso** and **Random forest**. These methods are described in section 4.7 while section 4.3.2 provides an overview of the supported the regression labels.

For **Linear** regression, the user can configure the fit intercept and normalize values. The configuration options are shown in figure 19.

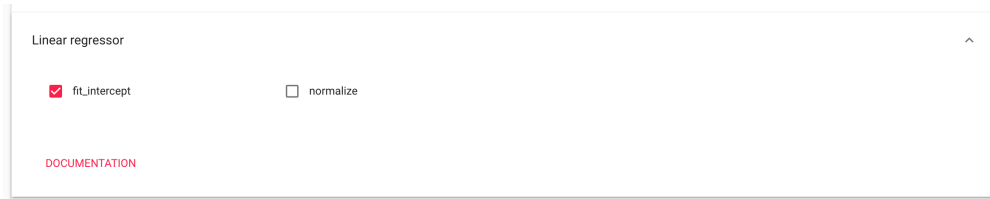
A screenshot of the 'Linear regressor' configuration panel. It features a title bar with 'Linear regressor' and a close button. Below the title bar, there are two checkboxes: 'fit\_intercept' which is checked with a red square, and 'normalize' which is unchecked with a grey square. At the bottom left, there is a red link labeled 'DOCUMENTATION'.

Figure 19: Linear regression configuration options

For **Lasso**, the user can configure the alpha, fit intercept and normalize values. The configuration options are shown in Figure 20.

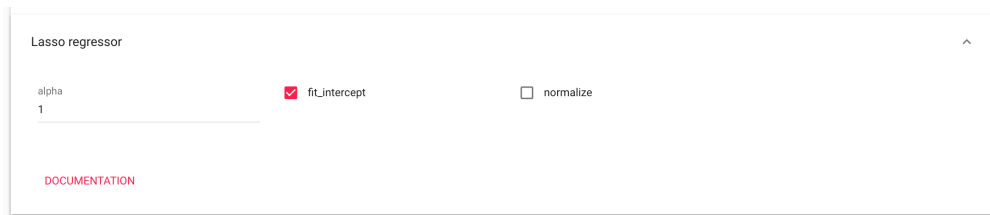
A screenshot of the 'Lasso regressor' configuration panel. It features a title bar with 'Lasso regressor' and a close button. Below the title bar, there is a text input field for 'alpha' with the value '1'. To the right of the input field are two checkboxes: 'fit\_intercept' which is checked with a red square, and 'normalize' which is unchecked with a grey square. At the bottom left, there is a red link labeled 'DOCUMENTATION'.

Figure 20: Lasso configuration options

For **Random Forest**, the user can configure the number of estimators, the maximum depth and the maximum number of features. The configuration options are shown in Figure 21.

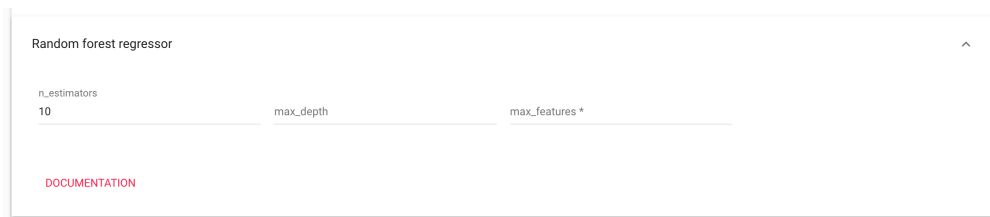
A screenshot of the 'Random forest regressor' configuration panel. It features a title bar with 'Random forest regressor' and a close button. Below the title bar, there are three text input fields: 'n\_estimators' with the value '10', 'max\_depth', and 'max\_features \*'. At the bottom left, there is a red link labeled 'DOCUMENTATION'.

Figure 21: Random forest configuration options

### 5.6.3 Clustering methods

The user can choose between using no clustering and the **k-means** clustering method. The clustering options are described in section 4.5.

The application provides additional configuration options for k-means, which are shown in Figure 22. The user can configure the number of clusters to create, the maximum number of iterations and the k-means algorithm.

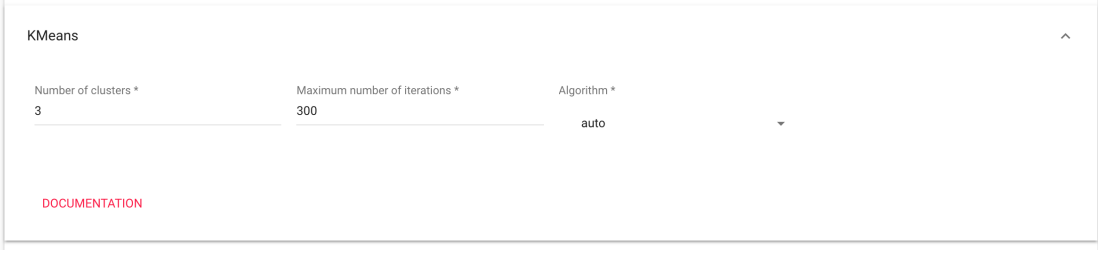
The image shows a web-based configuration form for the KMeans algorithm. The form is titled "KMeans" in the top left corner. It contains three input fields: "Number of clusters \*" with a value of "3", "Maximum number of iterations \*" with a value of "300", and "Algorithm \*" with a dropdown menu showing "auto". Below these fields is a red link labeled "DOCUMENTATION". The form has a clean, minimalist design with a light gray border and a white background.

Figure 22: K-means configuration options

### 5.6.4 Encoding methods

The application supports five encoding methods: simple index, boolean, frequency, last payload and complex index encoding. The encoding process can be further configured by specifying the padding, prefix length and task generation type. The encoding methods and options are described in section 4.2. The configurations options can be seen on the training page in Figure 15.

### 5.6.5 Labelling

The label is the value that the machine learning methods will try to predict. The application supports remaining time and trace numerical attribute labels for regression methods. For classification methods, the supported labels are duration, next activity, trace numerical and string attributes. For numerical classification tasks, such as duration and trace numerical attribute, the user must also choose a threshold for how to classify the label. These options are discussed in section 4.3. An example of classification numerical attribute label selection is shown in Figure 23.

For classification tasks, the label distribution can be tested and examined on the Labelling page.

Labelling

When using duration, the threshold is an integer in seconds. If the remaining time is below this threshold it is classified as `True` or `Fast`. Times above the threshold are classified as `False` or `Slow`.

Number attributes below the threshold are set as `True`.

Label type

Trace number attribute

Binary classification

Attribute name

AMOUNT\_REQ

First trace value: 20000

Threshold type

Custom

Use the threshold value below

Threshold

50000

Figure 23: Trace numerical attribute classification options

### 5.6.6 Temporal and inter-case features

The user can add a total of five features to the encoded log file, as can be seen from Figure 24. The features are discussed in section 4.4.

Temporal and intercase features

Temporal features

Temporal features are included as a numeric column. They represent time in seconds.

☐ Add remaining time
 ☐ Add elapsed time

Intercase features

These options use aggregated metrics of the entire log to create the value. The metrics are visible on the Logs page. For any event at a prefix length, the value is the count of metric on the date of the event.

☐ Add executed events
 ☐ Add resources used
 ☐ Add new traces

Figure 24: Temporal and inter-case features options

### 5.6.7 Hyperparameter optimization

Determining the most suitable input parameters for the previously described classification and regression methods can be difficult. By enabling hyperparameter optimization, the application will try to find the best parameters automatically. The user can configure the number of evaluation runs and the target performance metric, as can be seen in Figure 25.

Hyperparameter Optimization

Hyper optimization tries to guess the optimal method configuration to achieve the best performance metric value. It tries to optimize all the method values visible here. Maximum evaluations should be high to achieve the best result.

☐ Enable Hyper optimization

Number of evaluation runs \*

10

Performance metric \*

[DOCUMENTATION](#)

Figure 25: Hyperparameter optimization options

## 5.7 Validation page

The results of the completed classification and regression tasks are visible on the Validation page. Results can be filtered by the prediction, clustering, encoding, learning algorithm, padding and labelling configurations. An example of classification task filters is shown on Figure 26. The page also features a table with the full configuration of each prediction task. The results are visualized in a data table, a line chart and four bubble charts.

Validation selection

Split #5, repairExample.xes

Prediction method

☐ Regression  
Numeric values

☒ Classification  
Categorical values

Encoding methods

☒ Simple index
☒ Boolean
☒ Frequency
☒ Complex
☒ Last payload

Clustering methods

☒ None
☒ K-means clustering

Encoded log padding

☒ No padding
☐ With 0 padding

Classification methods

☒ KNN
☒ Decision tree
☒ Random forest

Prefix lengths

☒ 1
☒ 2
☒ 3
☒ 4
☒ 5
☒ 6
☒ 7
☒ 8
☒ 9
☒ 10
☒ 11
☒ 12
☒ 13
☒ 14
☒ 15
☒ 16
☒ 17
☒ 18
☒ 19

Label controls

Label type

Trace duration

Binary classification

Threshold type

Label mean

Threshold is label mean

Threshold

Figure 26: Validation page result filters

37

### 5.7.1 Classification results

The results for the filtered classification tasks are presented in a sortable data table. The contents of this table can be exported in a CSV format. Each row in the table represents a classification task with the task id, a string identifier of the task and the prefix length. The results of the training task are presented with the F1 score, accuracy, AUC, precision and recall. For binary classification tasks, the true positive, true negative, false positive and false negative metrics are also presented in the table. Figure 27 shows the classification results table.

ID	Task identity	F1 score	Accuracy	AUC	Prefix length	Precision	Recall	True positive	True negative	False positive	False negative
19	knn_simpleIndex_noCluster	0.741	0.588	0.5	1	0.588	1	130	0	91	0
20	knn_simpleIndex_noCluster	0.741	0.588	0.5	2	0.588	1	130	0	91	0
21	knn_simpleIndex_noCluster	0.741	0.588	0.5	3	0.588	1	130	0	91	0
22	knn_simpleIndex_noCluster	0.741	0.588	0.582	4	0.588	1	130	0	91	0
23	knn_simpleIndex_noCluster	0.547	0.566	0.612	5	0.707	0.446	58	67	24	72
24	knn_simpleIndex_noCluster	0.673	0.566	0.603	6	0.604	0.762	99	26	65	31
25	knn_simpleIndex_noCluster	0.601	0.561	0.606	7	0.646	0.562	73	51	40	57
26	knn_simpleIndex_noCluster	0.749	0.62	0.636	8	0.613	0.962	125	12	79	5
27	knn_simpleIndex_noCluster	0.823	0.76	0.863	9	0.728	0.946	123	45	46	7
28	knn_simpleIndex_noCluster	0.545	0.583	0.637	10	0.652	0.469	15	20	8	17

Figure 27: Classification results table

The classification tasks can be grouped together by run configuration to highlight the differences at every prefix length. This configuration is visualized in a line chart with the prefix length on the x-axis. The user can select any metric in the results table to be shown on the y-axis. Figure 28 shows the precision of various task configurations by prefix length.

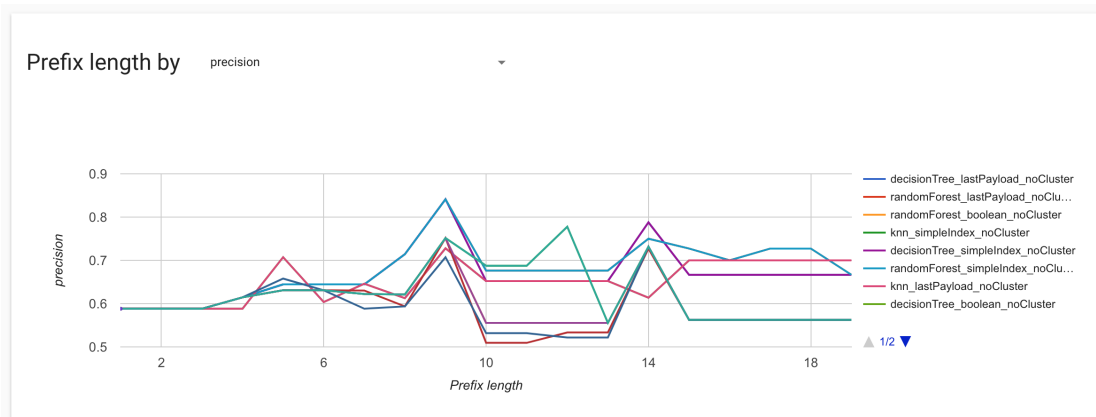


Figure 28: Precision by prefix length

All classification results are presented in four different bubble charts, as shown in Figure 29. These charts highlight the differences by classification method, clustering method, encoding method and prefix length. In all of the charts, the F1 score is the x-axis, the accuracy is the y-axis and the size of the bubble is determined by the AUC. The value on the bubble chart is the task id.

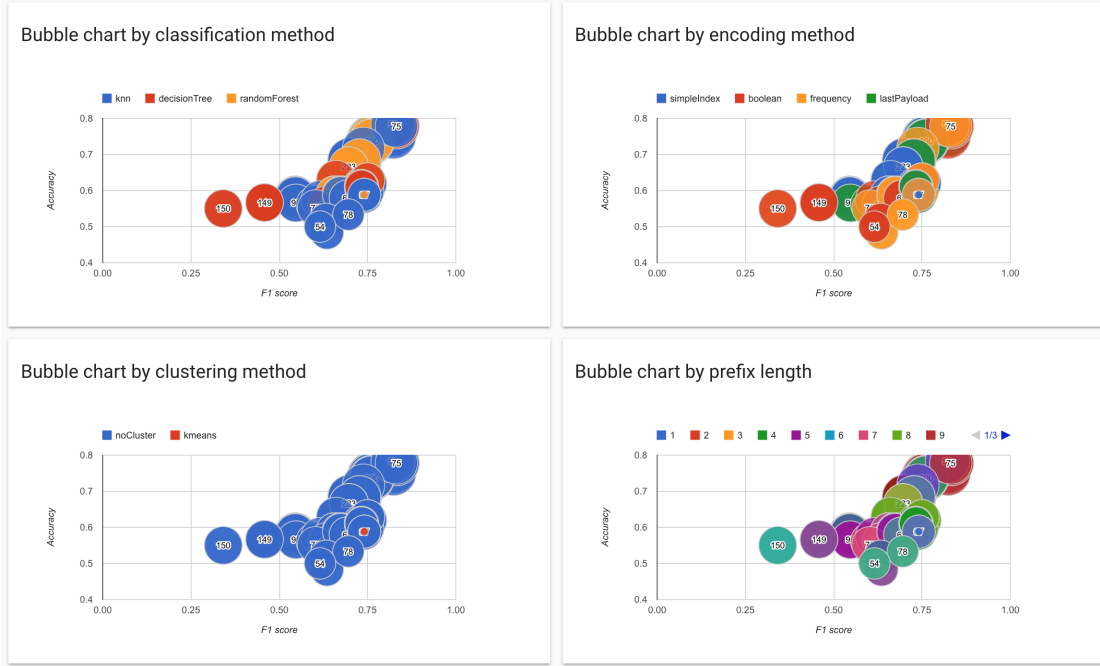


Figure 29: Classification bubble charts

### 5.7.2 Regression results

The results for the filtered regression tasks are presented in a sortable data table. The contents of this table can be exported in a CSV format. Each row in the table represents a regression task with the task id, a string identifier of the task and the prefix length. The results of the training task are presented with the MAE, RMSE and rscore metrics. Figure 30 shows the regression results table.

regression results [DOWNLOAD TABLE DATA](#)

ID	Task identity	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)	R-score	Prefix length
773	randomForest_simpleIndex_noCluster	0.257	0.316	-0	1
774	randomForest_simpleIndex_noCluster	0.257	0.316	-0	2
775	randomForest_simpleIndex_noCluster	0.255	0.315	-0	3
776	randomForest_simpleIndex_noCluster	0.239	0.284	0.006	4
777	randomForest_simpleIndex_noCluster	0.225	0.29	0.089	5
778	randomForest_simpleIndex_noCluster	0.18	0.265	0.226	6
779	randomForest_simpleIndex_noCluster	0.175	0.26	0.158	7
780	randomForest_simpleIndex_noCluster	0.168	0.25	0.288	8
781	randomForest_simpleIndex_noCluster	0.052	0.122	0.835	9
782	randomForest_simpleIndex_noCluster	0.195	0.239	0.07	10

1 2 10 30 38

Figure 30: Regression results table

The regression tasks can be grouped together by run configuration to highlight the differences at every prefix length. This configuration is visualized in a line chart with the the prefix length on the x-axis. The user can select any metric in the results table to be shown on the y-axis. Figure 31 shows the precision of various task configurations by prefix length.

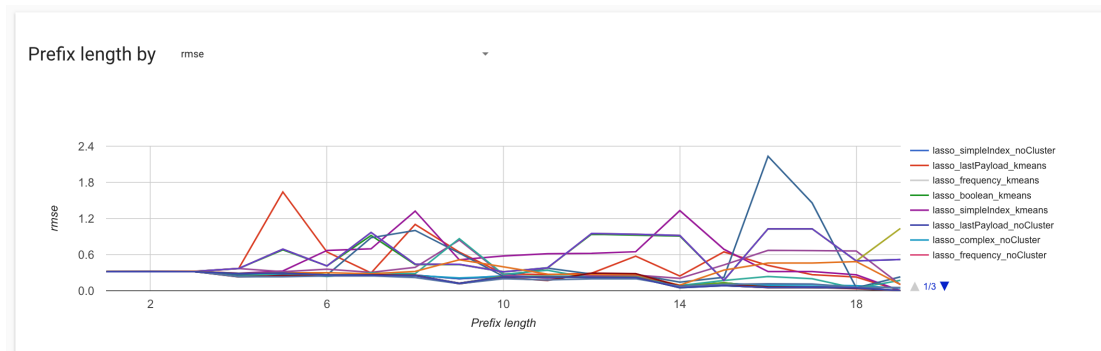


Figure 31: RMSE by prefix length

All regression results are presented in four different bubble charts, as shown in Figure 29. These charts highlight the differences by regression method, clustering method, encoding method and prefix length. In all of the charts, the MAE is the x-axis, the RMSE is the y-axis and the size of the bubble is determined by the rscore value. The number on the bubble chart is the task id.



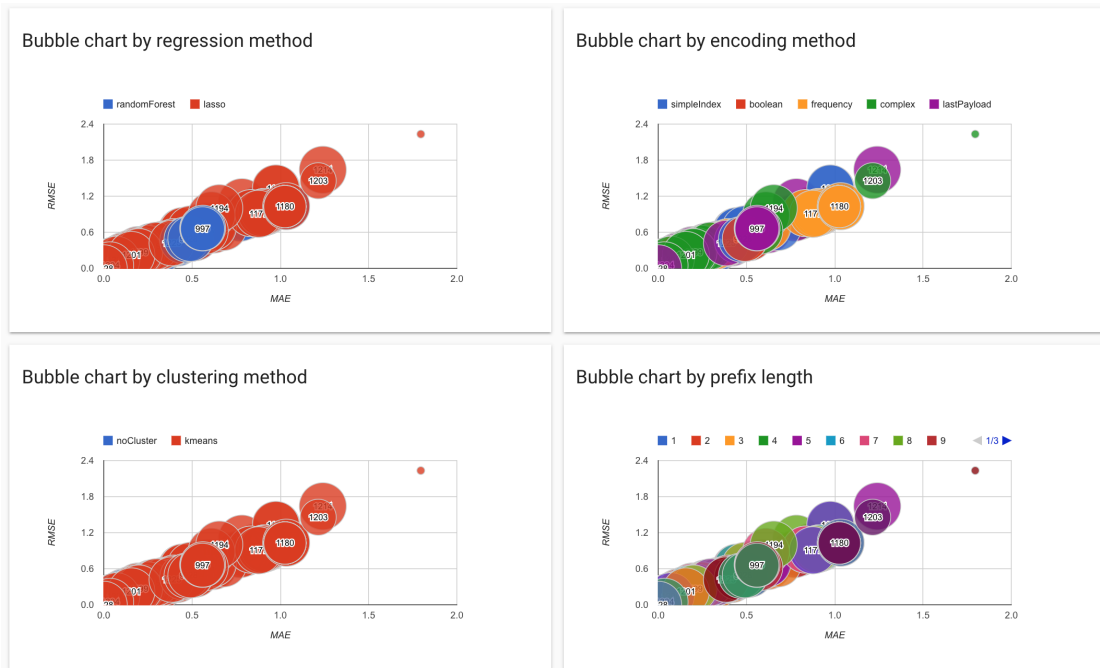


Figure 32: Regression bubble charts

## 6 Tool implementation

This section provides an overview of the architecture of Nirdizati Research, the used technologies and the development process.

### 6.1 Architecture

Nirdizati Research is an improved version of Nirdizati Training and as such has a similar architecture. Nirdizati Research retains the concept of a front-end application for prediction task management, a back-end web application that provides the data for the front-end and a queuing system that manages the worker applications. Figure 33 shows an overview of the application architecture.

The communication between the front and back-end applications is achieved in the JSON (JavaScript Object Notation) [18] format.

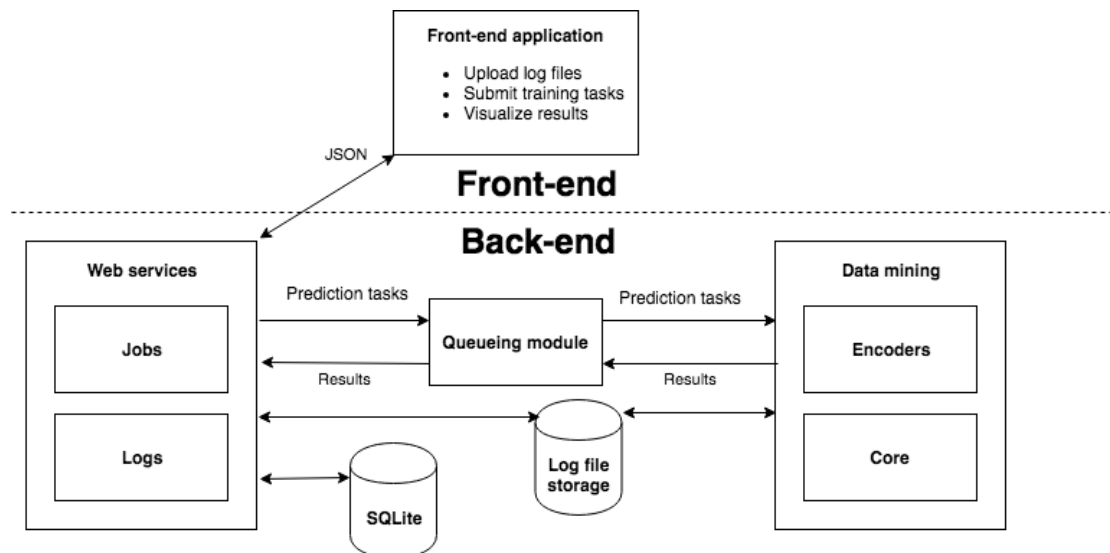


Figure 33: Nirdizati Research architecture

The back-end application modules can be divided into web-service and data-mining modules. The web-service modules are mainly concerned with processing the queries from the front-end while also being the only modules that have direct access to the database. This separation of concerns makes it possible to run the prediction tasks directly without the use of web services or database objects.

The data-mining modules serve as the "client" of the queuing system, and by proxy, of the web-service modules. The intent was that the data-mining modules would have no dependencies on the surrounding web application and they could be directly ported to other applications without any changes. In practice this means

that the data-mining modules only comprise of pure functions that return plain objects.

### **6.1.1 Back-end architecture**

The back-end application comprises of 5 main modules in addition to the database and storage. Core and Encoders are classified as data-mining modules while Jobs and Logs are web-service modules. The Queuing module serves as the middleware between the modules.

The Core module is the main module of the application. It includes the prediction methods for classification and regression. It is also responsible for calculating the various evaluation metrics based on the prediction results.

The Encoders module is one of the data-mining modules and it encodes and labels the log for further use in the Core module. The Jobs module provides the REST endpoints for creating and managing the prediction tasks. This module will also add prediction tasks to the worker queues.

The Logs module is responsible for log upload and storage. It provides the REST endpoints for creating, listing and querying information about log and split configurations.

The Queuing module manages the worker applications and uses the Core and Encoding modules to complete the predictions tasks.

The log files are stored on the file system. The log metrics, split configurations and prediction task results are stored in a database.

### **6.1.2 Front-end architecture**

The front-end application is implemented as a SPA (Single Page Application). Instead of retrieving the entire web page from the server, a SPA rewrites the current page based on user interactions. Similar to a desktop application, all the required user interface code is retrieved on the first page load. Additional resources, such as the data describing the state of the prediction tasks, is dynamically loaded when necessary.

The main functions of the front-end are the uploading of log files, configuration of the training tasks, overview of the running training tasks and visualization of the results.

## 6.2 Technologies

Nirdizati Training is the foundation on which Nirdizati Research is built and both application share a similar architecture and process flow. However, there are significant differences. The backend of Nirdizati Training was written in Python 2, but the use of the OpyenXES [19] package for log encoding required the use of Python 3. Therefore the backend of Nirdizati Research was rebuilt from scratch to take into account the added functionalities. To accompany the additional functionalities, Nirdizati Research also introduces an entirely new frontend application.

### 6.2.1 Back-end technologies

The back-end web application is written in Python and supports Python 3.5, 3.6 and the development branch of 3.6. The web application is built using the Django Framework<sup>11</sup> version 1.11.7 and the web service API is built using the Django REST Framework<sup>12</sup> version 3.7.1.

The queuing system is retained from Nirdizati Training [14]. Each received prediction task is put into a worker queue, which allows the application to process multiple prediction tasks in parallel. The queuing system is implemented with Django-RQ<sup>13</sup>.

OpyenXES [19] is a Python package for handling event logs based on the XES standard and it is based on the Java implementation OpenXes. This package allows the application to parse event logs in XES and MXML format. Furthermore, it is used to extract event attributes and generate the various log metrics.

The aggregated log metrics, split configurations and prediction task validation results are stored in an SQLite<sup>14</sup> database. SQLite [20] was chosen due to its compact size and good integration with Django.

The Hyperopt package is used for managing the Hyperparameter optimization tasks<sup>15</sup>.

---

<sup>11</sup><https://www.djangoproject.com/>

<sup>12</sup><http://www.django-rest-framework.org/>

<sup>13</sup><https://github.com/ui/django-rq>

<sup>14</sup><https://www.sqlite.org/index.html>

<sup>15</sup><https://github.com/hyperopt/hyperopt>

### 6.2.2 Front-end technologies

The frontend application of Nirdizati Research is built using React<sup>16</sup>, which is a JavaScript library for building user interfaces. The state of the application is managed with Redux<sup>17</sup>, which is a predictable state container for JavaScript apps. React and Redux were chosen due to the developer's prior experience with the libraries.

The application is styled with react-md<sup>18</sup>, which provides React components in Material Design<sup>19</sup>. The charts in the application are generated using Google Charts<sup>20</sup>. The choices for styling and charting were influenced by their prior usage in Nirdizati Training.

### 6.2.3 Development process

Both the frontend and backend applications are partially covered by unit tests to increase application reliability and to make it more suitable for future development. The backend application is tested using the Python standard unittest<sup>21</sup> framework. Both applications use Travis CI for continuous testing<sup>22 23</sup>. After testing the frontend application, the test coverage metric is calculated, which is at 74% of all lines of code [21].

---

<sup>16</sup><https://reactjs.org/>

<sup>17</sup><https://redux.js.org/>

<sup>18</sup><https://react-md.mlaursen.com/>

<sup>19</sup><https://material.io/>

<sup>20</sup><https://developers.google.com/chart/>

<sup>21</sup><https://docs.python.org/3/library/unittest.html>

<sup>22</sup><https://travis-ci.org/TKasekamp/predict-python>

<sup>23</sup><https://travis-ci.org/TKasekamp/predict-react>

## 7 Evaluation and Comparison

### 7.1 Evaluation

In this section, we evaluate the application by using a real-life event log pertaining to the treatment of sepsis cases in a hospital [10]. The aim is to find the best predictive model for predicting the remaining time and the next activity. Another aim of the evaluation is to determine if inter-case features and hyperparameter optimization can improve the predictive model.

The hospital event log reference [10] shows that the each trace has an average of 14.49 events. Therefore in the following examples we are going to generate prediction tasks up to prefix length 20 as this will cover most of the relevant cases. The event log was split sequentially with a training/test split of 80%/20%.

#### 7.1.1 Remaining time prediction

As a starting point for finding the best model for remaining time prediction, we generated tasks using every combination of the three regression, five encoding and two clustering methods. These tasks were created with no padding and with every prefix length up to 20. In total, 600 prediction tasks were created.

Using the filtering options, the search for the best predictive model can be narrowed down to configurations using no clustering and lasso regressor with every encoding method. Figure 34 shows the best five configurations for remaining time prediction by prefix length. Of these configurations, the models using simple index encoding perform the worst, if measured using RMSE. The other four encoding methods perform rather similarly and the final choice for the best method would depend on the prefix length to optimize the model for.

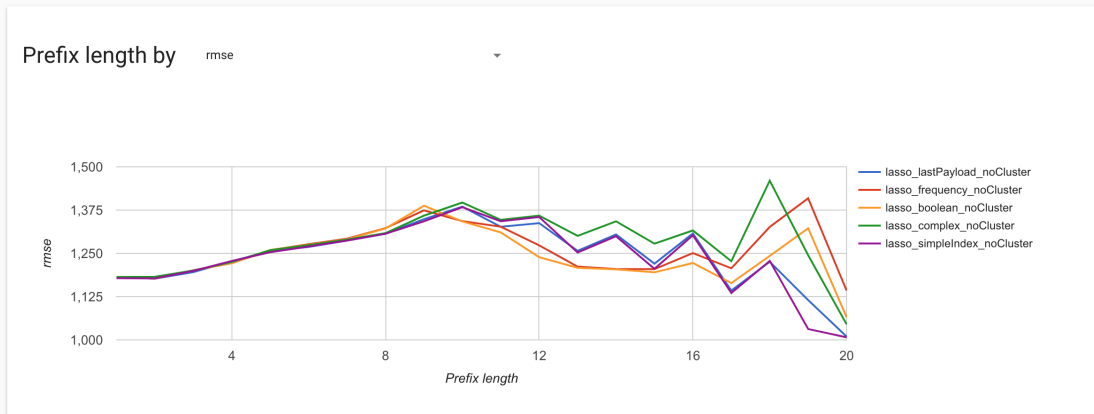


Figure 34: Remaining time prediction configurations

To evaluate if inter-case features improve the model quality, we have selected prefix length 16 as the test focus. As can be seen from Figure 34, the best performing configuration at prefix 16 is lasso regressor with complex index encoding and no clustering. Figure 35 shows the results for prediction task configurations where Task ID 1073 has no additional features and Task ID 1240 has the added inter-case features of executed events, resources used and new traces. The Figure shows that adding these features decreases the error measured by RMSE and MAE. Therefore, adding inter-case features can at times improve the predictive model.

ID	Task identity	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)	R-score	Prefix length
1073	lasso_complex_noCluster	933.938	1,316.139	-0.104	16
1240	lasso_complex_noCluster	918.62	1,309.299	-0.093	16

Figure 35: Remaining time prediction with and without inter-case features

However, finding a prefix at which inter-case features improve the model is a difficult task which requires substantial experimentation with various parameters. Additional features are not always guaranteed to improve the predictive model.

### 7.1.2 Next activity prediction

As a starting point for finding the best model for next activity prediction, we generated tasks using every combination of the three classification, five encoding and two clustering methods. These tasks were created with no padding and with every prefix length up to 20. In total, 600 prediction tasks were created.

Figure 36 shows all prediction model configurations by prefix length and F1-score. As there are 30 unique configurations, the chart does not allow us to easily distinguish the best predictive model. However, this array of configurations can be filtered down to a more manageable size. Figure 37 shows the four candidates for the best next activity prediction configuration. All four remaining configurations use a combination of random forest and decision tree classifiers with simple index and frequency encoding with no clustering. The final choice for the best method would depend on the prefix length to optimize the model for.

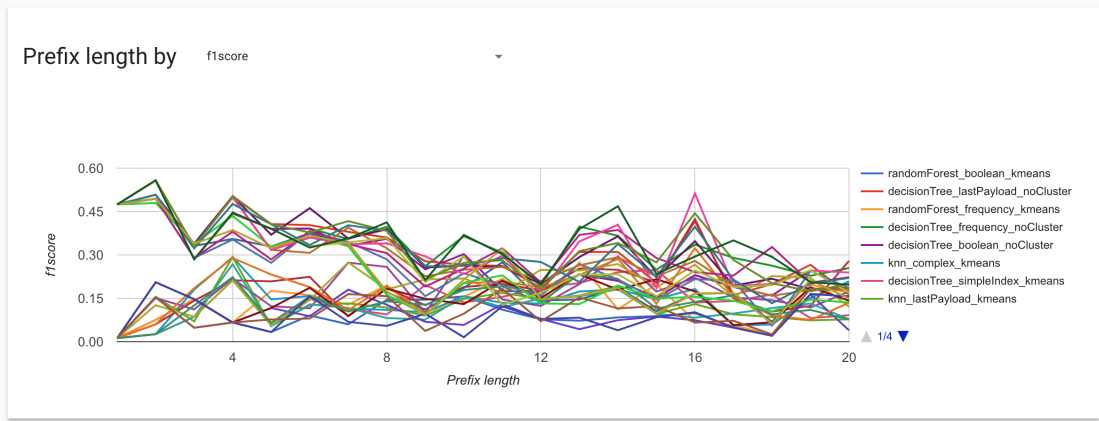


Figure 36: All next activity predictions

To evaluate if hyperparameter optimization can improve the model quality, we have selected prefix length 15 as the test focus. As can be seen from Figure 37, the best performing configuration at that prefix is decision tree classifier with frequency encoding and no clustering. Figure 38 shows the results for prediction task configurations where Task ID 311 used the default parameters for decision tree and Task ID 1242 has been optimized for F1-score. The Figure shows that the F1-score, Accuracy, Precision and Recall metrics have all been improved over the default configuration. Figure 38 also indicates that hyperparameter optimization is a viable option for improving the predictive model.



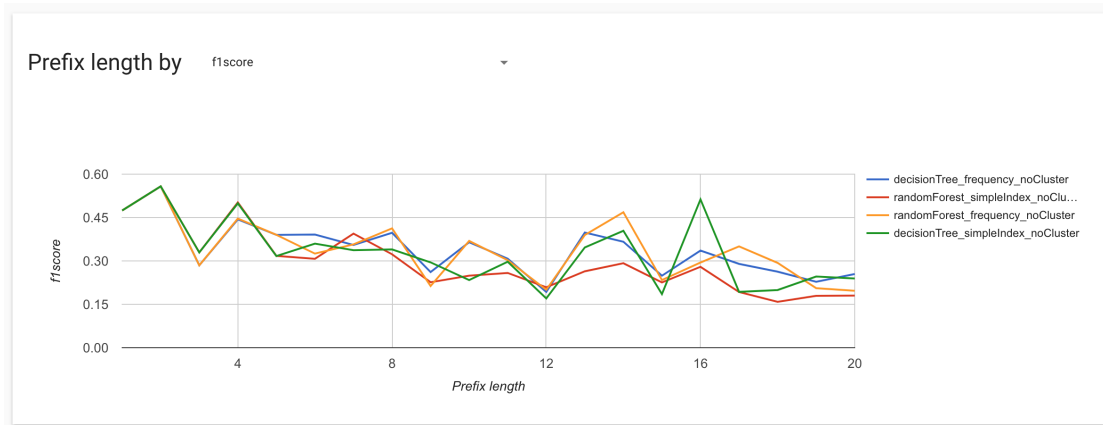


Figure 37: Filtered next activity predictions

ID	Task identity	F1 score ▼	Accuracy	AUC	Prefix length	Precision	Recall
1242	decisionTree_frequency_noCluster	0.334	0.479	0	15	0.342	0.336
311	decisionTree_frequency_noCluster	0.249	0.411	0	15	0.246	0.259

Figure 38: Next activity prediction with and without hyperparameter optimization

## 7.2 Performance

This section provides a performance overview of the encoding and machine learning methods of Nirdizati Research. The following tests are intended as a guide about the relative performance of Nirdizati Research as several hundred prediction tasks can be submitted simultaneously. Therefore it is important to understand how different configurations can impact the time it takes to complete the prediction tasks.

Several of the tests in this section were completed using a real-life log file pertaining to a loan application process in a financial institution. This log was presented in the 2017 BPI challenge [22]. This log file was chosen for its large size as it contains 31,509 traces and 1,202,267 events.

The tests were conducted on a 2015 MacBook Pro with the following configuration:

- **CPU:** 2.7 GHz Intel Core i5 4 core CPU
- **RAM:** 8 GB 1867 MHz DDR3
- **Operating system:** macOS High Sierra 10.13.4

The computer was not used for resource-intensive tasks during the execution of the tests.

### 7.2.1 Encoding methods

The first test is a comparison of the five encoding methods. As the focus of this test is on the encoding and labelling part of the application, these numbers do not include the time to load and parse the log from the file system as this would be identical for all five methods. These tests were conducted with the BPI Challenge 2017 event log [22]. To create the largest possible encoded data set, the test configuration was the maximum prefix length of 180 with zero padding and no labelling.

The results are displayed in Table 18. Boolean and frequency encoding are the fastest as they only include **Boolean** or **Integer** values to the encoded log instead of the more computationally complex structure **String**, as it is done in the other three encoding methods. Last payload and simple index encoding are similar in performance. Complex index encoding takes by far the longest as it has to include all attributes for every event.

Encoding method	Time in seconds
Complex	440.85
Last payload	58.91
Simple index	52.37
Frequency	29.64
Boolean	27.26

Table 18: Encoding method performance comparison

### 7.2.2 Log size impact on encoding performance

The encoding performance might be influenced by the number of traces in the event log. The following is a test of every encoding method with three log files of various size. The first log file is about a treatment process from a hospital [10], which contains 1050 traces and 15,214 events in total. The second log file is a real-life log from a financial institution and it was introduced for the 2012 BPI Challenge [23]. It contains 13,087 traces and 262,200 events in total. The largest log file in the test was introduced for the 2017 BPI Challenge [22] and contains 31,509 traces and 1,202,267 events.

The test measures only the encoding of the log file without the time taken to read in and parse the log file from the file system. The test configuration was prefix length 20 with zero padding.

As can be seen from Table 19, the size of the event log has an effect on the encoding time. For a log file with only a 1000 traces such as the hospital log, the differences in encoding methods are not significant. For a large log file such as BPI

Encoding method	Hospital log (seconds)	BPI 2012 (seconds)	BPI 2017 (seconds)
Simple index	0.21	2.86	14.60
Boolean	0.08	1.71	7.29
Frequency	0.11	1.35	5.44
Complex	0.50	4.32	45.27
Last payload	0.20	2.57	20.11
Read in time	3.01	45.36	323.07

Table 19: Encoding method performance by log size

Challenge 2017, the choice of encoding method greatly impacts the encoding time. It can also be observed that the number of traces in the log file has an enormous impact on the time it takes to read and parse the log file.

### 7.2.3 Machine learning methods

The aim of the following test is to give an overview of how long it takes to generate a prediction model with every regression and classification method. These tests were conducted with the BPI Challenge 2017 event log [22]. This process includes reading in the log file from the file system, encoding, labelling, applying the machine learning method and calculating the results. For both regression and classification, the training and test set were split sequentially with the split percentage of 80%/20%. The resulting training and test set contained 23,898 and 5942 rows respectively.

The classification tests were run with the configuration: encoding type boolean, no padding, prefix length 20, prediction label duration, no clustering and default classification method parameters. The results of the test can be seen in Table 20 with a separation of the task stages. It can be seen that decision tree is the fastest while KNN is the slowest classification method. However, the difference between the methods is relatively small compared to the overall time of the test.

Classification method	Log read in time (seconds)	Encoding (seconds)	Method time (seconds)	Total (seconds)
Decision tree	278.42	1.66	<b>0.06</b>	297.94
Random forest	268.41	1.78	<b>0.39</b>	293.70
KNN	286.33	1.19	<b>4.60</b>	317.56

Table 20: Classification method performance comparison

The regression tests were run with the configuration: encoding type boolean,

no padding, prefix length 20, prediction label remaining time, default regression method parameters and no clustering. The results of the test can be seen in Table 21 with a separation of the task stages. With this configuration, linear is the fastest method while random forest is 0.24 seconds slower. However, the time for any regression method takes up an insignificant percentage of the total task time.

Regression method	Log read in time (seconds)	Encoding (seconds)	Method time (seconds)	Total (seconds)
Linear	294.86	1.88	<b>0.09</b>	345.08
Lasso	298.27	1.78	<b>0.15</b>	334.85
Random forest	232.40	1.19	<b>0.35</b>	254.71

Table 21: Regression method performance comparison

The tests in Tables 20 and 21 also show that reading in the log file from the file system is the most time-consuming stage of the entire model generation process. It can also be seen that the time for the log read in and encoding steps differ across all test runs, suggesting that the background processes of the test computer can influence these times.

#### 7.2.4 Hyperparameter optimization

Compared to a normal prediction task, the machine learning method is invoked multiple times with various parameters when using Hyperparameter optimization. In this test, we compared the results of a classification prediction task with and without Hyperparameter optimization. The optimization task was evaluated 10 times with the performance metric **F1-score**.

The test was conducted with the BPI Challenge 2017 event log [22]. The test configuration was encoding method simple index, no padding, prefix length 20, prediction label next activity, default classification method parameters and no clustering.

As can be seen from Table 22, enabling the optimization increases the task execution time by 50 seconds while improving the F1-score metric by close to 1%. This makes hyperparameter optimization a credible option for improving the prediction model.

	Time in seconds	F1-score
With optimization	377.61	0.4649
No optimization	326.09	0.4560

Table 22: Hyperparameter optimization comparison

### 7.2.5 Comparison with Nirdizati Training

Nirdizati Training is a similar web application for finding the best predictive model for an event log. In this test, we compared the encoding performance of Nirdizati Research and Nirdizati Training. Only the encoding performance was measured as the machine learning methods in both applications are identical. The event log for this test is a real-life event log about sepsis cases from a hospital [10]. This log contains 1050 traces and 15214 events.

The test measured reading in the log file from the file system and the encoding. Both applications were tested with three encoding methods: simple index, boolean and frequency as complex and last payload were not included in the final released version of Nirdizati Training <sup>24</sup>. Nirdizati Training offered no additional encoding configuration options, therefore the configuration of Nirdizati Research was used to replicate the encoded log file. The configuration was as follows: prefix length at the log maximum of 185, zero padding enabled, all prefixes in a single log file.

Encoding method	Nirdizati Research	Nirdizati Training
Simple index	17.17	46.07
Boolean	5.02	373.66
Frequency	4.93	374.36

Table 23: Encoding method comparison

As can be seen from Table 23, the encoding methods offered by Nirdizati Research are considerably faster. This is because Nirdizati Research uses the OpyenXES [19] package for handling event logs while Nirdizati Training needs to convert the event log to a CSV file before it can be encoded.

---

<sup>24</sup><https://github.com/nirdizati/nirdizati-training-backend>

## 7.3 Comparison with Nirdizati Training

Nirdizati Research is the tool presented in this thesis, but it shares some of the approaches with its predecessor Nirdizati Training. This section aims at giving an overview of their differences, but it does not mention the common functions.

The improvements in Nirdizati Research fall into two broad categories: new features and improvements to existing features.

### 7.3.1 New features

Nirdizati Research introduces several new features related to log file handling, labelling and advanced configuration.

Log files can now be in XES, MXML, gzipped XES and MXML while Nirdizati Training only allowed the XES format. Nirdizati Research allows to upload the test and training set separately. When uploading a single log file, the user can select the test/training set split percentage. There are also four options for how the log should be split.

Nirdizati Training provided 3 classification, 4 regression and 1 clustering methods, however these methods were used with their default configuration. For prediction fine-tuning, Nirdizati Research adds the option to configure a subset of the method parameters.

Nirdizati Training added the remaining time of the trace to the encoded log file. Nirdizati Research builds on this by adding elapsed time and three inter-case features.

Nirdizati Training had the option to use regression methods to predict the remaining time and to use classification methods to predict the remaining time and duration. Nirdizati Research adds the choice to label with a trace numerical attribute for regression. For classification, the user can label with the duration, next activity, trace numerical and string attributes.

To see the distribution of labels prior to applying machine learning methods, Nirdizati Research adds the option to test out labelling configurations.

Hyperparameter optimization was also added to help find the best configuration for a predictive model.

### 7.3.2 Improvements

The main improvements are evident in the performance of the encoding process, as can be seen in Table 23. Nirdizati Research adds the option to encode each prefix to separately, to choose the prefix length and to configure the log padding.

As Nirdizati Research adds many more configuration options, one of the focuses of development was the ability to filter the prediction results. The trends of the various configurations can be seen on a line chart that can display any metric by

prefix length. Also added is a bubble chart that depicts prediction tasks by prefix length.

The recall and precision metrics are now calculated for classification. Furthermore, the binary classification tasks include the confusion matrix metrics shown in Table 17.

Both the frontend and backend applications were completely rebuilt to support the added functionality. The frontend is an entirely new application while the backend has few lines of code in common with Nirdizati Training. Code quality was further improved by adding automated unit tests to the applications. The result of these improvements is a faster and more streamlined user experience. The architecture and technologies used for Nirdizati Research should allow for future development without needing to rebuild the entire application from scratch.

## 8 Conclusion

This thesis introduces a tool called Nirdizati Research for creating and validating models for predictive process monitoring. The application allows the user to create a predictive model for remaining time, duration, next activity and trace attribute prediction. The user can select among several encoding, clustering and machine learning method combinations. Hyperparameter optimization was also added to improve the quality of the models.

Nirdizati Research was evaluated using a real-life event log. Furthermore, this thesis gave overview of the performance impact of various encoding and machine learning methods. Finally, it compared the functionality of the presented application Nirdizati Research against a similar tool Nirdizati Training.

Finding the best predictive model for a given event log requires trying out various configurations and comparing hundreds of possible model candidates. The goal of the presented tool is to make this process easier.

The most significant area for future improvement is the ability to use the created models during runtime prediction and the work to integrate this feature is currently on-going. At the time of writing some of the required functionality is already present in the backend source code repository.

While there are no significant shortcomings, the optimization of the encoding and prediction task performance was not a development focus. Therefore, a major area of improvement could be the overall performance of the application.

The validation page could be improved by adding more or reworking the filtering options. While the current options work, the user experience for searching through the models could be more intuitive. Furthermore, additional visualization options could be added that better highlight the differences of the model results.

Hyperparameter optimization can currently only find the best parameters for the three classification and three regression methods. This could be built upon by also finding the best parameters for every combination of clustering, labelling, encoding configuration and additional feature columns. By optimizing everything, the process to find the best predictive model would be automatic and therefore faster for the user.



## References

- [1] W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer Berlin Heidelberg, 2016.
- [2] Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, and Fredrik Milani. “Predictive Process Monitoring Methods: Which One Suits Me Best?” In: *CoRR* abs/1804.02422 (2018). arXiv: 1804.02422. URL: <http://arxiv.org/abs/1804.02422>.
- [3] B. F. van Dongen, R. A. Crooy, and W. M. P. van der Aalst. “Cycle Time Prediction: When Will This Case Finally Be Finished?” In: *On the Move to Meaningful Internet Systems: OTM 2008*. Ed. by Robert Meersman and Zahir Tari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 319–336. ISBN: 978-3-540-88871-0.
- [4] W.M.P. van der Aalst, M.H. Schonenberg, and M. Song. “Time prediction based on process mining”. In: *Information Systems* 36.2 (2011). Special Issue: Semantic Integration of Data, Multimedia, and Services, pp. 450–475. ISSN: 0306-4379. DOI: <https://doi.org/10.1016/j.is.2010.09.001>. URL: <http://www.sciencedirect.com/science/article/pii/S0306437910000864>.
- [5] Andreas Rogge-Solti and Mathias Weske. “Prediction of Remaining Service Execution Time Using Stochastic Petri Nets with Arbitrary Firing Delays”. In: *Service-Oriented Computing*. Ed. by Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 389–403. ISBN: 978-3-642-45005-1.
- [6] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. “Predictive Monitoring of Business Processes”. In: *CoRR* abs/1312.4874 (2013). arXiv: 1312.4874. URL: <http://arxiv.org/abs/1312.4874>.
- [7] Marco Federici, Williams Rizzi, Chiara Di Francescomarino, Marlon Dumas, Chiara Ghidini, Fabrizio Maria Maggi, and Irene Teinemaa. “A ProM Operational Support Provider for Predictive Monitoring of Business Processes”. In: *BPM*. 2015.
- [8] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. “Time and Activity Sequence Prediction of Business Process Instances”. In: *CoRR* abs/1602.07566 (2016). arXiv: 1602.07566. URL: <http://arxiv.org/abs/1602.07566>.

- [9] Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, Giulio Petrucci, and Anton Yeshchenko. “An Eye into the Future: Leveraging A-priori Knowledge in Predictive Business Process Monitoring”. In: *Business Process Management*. Ed. by Josep Carmona, Gregor Engels, and Akhil Kumar. Cham: Springer International Publishing, 2017, pp. 252–268. ISBN: 978-3-319-65000-5.
- [10] F (Felix) Mannhardt. *Sepsis Cases - Event Log*. en. 2016. DOI: 10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460. URL: <https://data.4tu.nl/repository/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460>.
- [11] Anna Leontjeva, Raffaele Conforti, Chiara Di Francescomarino, Marlon Dumas, and Fabrizio Maria Maggi. “Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes”. In: *BPM*. 2015.
- [12] Chiara Di Francescomarino, Marlon Dumas, Marco Federici, Chiara Ghidini, Fabrizio Maria Maggi, and Williams Rizzi. “Predictive Business Process Monitoring Framework with Hyperparameter Optimization”. In: *Advanced Information Systems Engineering*. Ed. by Selmin Nurcan, Pnina Soffer, Marko Bajec, and Johann Eder. Cham: Springer International Publishing, 2016, pp. 361–376. ISBN: 978-3-319-39696-5.
- [13] Kerwin Jorbina. *A Web-Based Tool For Predictive Process Analytics*. University of Tartu. 2017.
- [14] Ayham Taleb. *A Web Tool For The Comparison Of Predictive Process Monitoring Algorithms*. University of Tartu. 2017.
- [15] Arik Senderovich, Chiara Di Francescomarino, Chiara Ghidini, Kerwin Jorbina, and Fabrizio Maria Maggi. “Intra and Inter-case Features in Predictive Process Monitoring: A Tale of Two Dimensions”. In: *Business Process Management*. Ed. by Josep Carmona, Gregor Engels, and Akhil Kumar. Cham: Springer International Publishing, 2017, pp. 306–323. ISBN: 978-3-319-65000-5.
- [16] Arik Senderovich, Matthias Weidlich, Avigdor Gal, and Avishai Mandelbaum. “Queue mining for delay prediction in multi-class service processes”. In: *Information Systems* 53 (2015), pp. 278–295. ISSN: 0306-4379.
- [17] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1996), pp. 267–288. ISSN: 00359246. URL: <http://www.jstor.org/stable/2346178>.
- [18] JSON. *Json home*. URL: <http://json.org>. (accessed: 22.03.2018).

- [19] H. Valdivieso, J. Lee, Arias M., Rojas E., and Sepúlveda Munoz-Gama J. *OpyenXes: A Complete Open-Source Python Library for the Extensible Event Stream Standard (under review)*. 2017.
- [20] *SQLite* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 9-May-2018]. 2018. URL: <https://en.wikipedia.org/w/index.php?title=SQLite&oldid=840189627>.
- [21] *TKasekamp/predict-react | Build #180 | Coveralls - Test Coverage History and Statistics*. [Online; accessed 21-May-2018]. 2018. URL: <https://coveralls.io/builds/17076511>.
- [22] B.F. Van Dongen. *BPI Challenge 2017*. en. 2017. DOI: 10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b. URL: <https://data.4tu.nl/repository/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b>.
- [23] B.F. Van Dongen. *BPI Challenge 2012*. nl. 2012. DOI: 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f. URL: <https://data.4tu.nl/repository/uuid:3926db30-f712-4394-aebc-75976070e91f>.

**Non-exclusive licence to reproduce thesis and make thesis public**

I, **Tõnis Kasekamp** (date of birth: 21st of April 1993),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
  - 1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
  - 1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

**A Web Application to Support Research in Predictive Monitoring Tasks**

supervised by Fabrizio Maria Maggi

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 21.05.2018